UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"DESARROLLO E IMPLEMENTACIÓN DE SISTEMAS WEB CON SOFTWARE LIBRE EN LINUX"

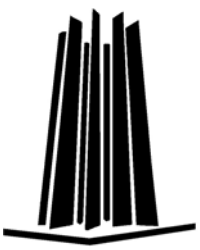
TRABAJO ESCRITO

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 :

ARREDONDO GUZMÁN MARIO ALBERTO

ASESOR: M. EN I. ELIO VEGA MURGUÍA



MÉXICO, 2007.

En agradecimiento a nuestra institución y a las personas que directa e indirectamente hicieron posible la realización de este trabajo.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN.

ÍNDICE

INTRODUCCIÓN

Enfocado a aquellos profesionales, que están en constante capacitación y búsqueda de tecnologías relevantes, con las cuales pueda optimizar sus habilidades, la Dirección General de Servicios de Cómputo Académico diseño el Diplomado llamado Diseño e implementación de sistemas con Software Libre en LINUX, que se imparte en el Centro de Extensión en Cómputo y Telecomunicaciones Mascarones.

Módulos:

1.- Sistema operativo Linux

2.- Instalación y administración de Linux

3.- Editores 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.- Desarrollo de aplicaciones Postgresql y PHP

OBJETIVO GENERAL

Se conocerán nuevas herramientas administrativas que 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.

1. Sistema Operativo Linux

1.1 Objetivo

Se instalará el sistema operativo Linux en PC's y se configurará la tarjeta de red, video y particiones del disco duro.

1.2 Introducción

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

En la década de los 60´s cuando todo el software se mantenía en una arquitectura cerrada dictada por el diseño de cada productor de hardware los Laboratorios Bell de AT&T en conjunto con GE trabajaron en el MIT en un proyecto llamado MULTICS (Servicio de computo de información Multiplexada) un antiguo sistema de tiempo compartido el cual tenía como finalidad proporcionar un soporte para diferentes arquitecturas de hardware compartiendo los recursos de hardware y software con la finalidad de eliminar el concepto de arquitectura cerrada.

En el año de 1969 el proyecto MULTICS fracasa por su alta demanda de espacio requerido en disco duro y memoria RAM, lo cual evita que el proyecto sea financiable, abandonándolo en este año. Kenneth Thompson y Dennis Ritchie, ingenieros de laboratorios BELL deciden continuar con el proyecto por su cuenta, desarrollando herramientas de programación que les permitieron tener un mejor desempeño de los programas diseñados por ellos mismos.

Kenneth Thompson desarrolla un lenguaje de programación de bajo nivel al cual le llamó lenguaje B, éste es analizado y mejorado por Dennis Ritchie, desarrollando un lenguaje de programación más potente y robusto al cual le llamaron C, el cual se ha mantenido desde entonces hasta la fecha como un lenguaje de programación de excelencia para los diseñadores de programas que tienen un alto nivel de interacción con el hardware.

Para el año de 1971 es lanzada la segunda versión de UNIX, esta versión se diseño en lenguaje C, lo que le permitió la portabilidad entre las diferentes arquitecturas de hardware, de este modo se inicia el diseño de la arquitectura abierta en los equipos de cómputo. En el año de 1983 Novell compra UNIX a AT&T y lo registra con el nombre de System V obteniendo todos los derechos sobre el producto.

El nombre de UNIX se desprende de una aberración de Multics, ya que Unix inicio como un proyecto menos ambicioso el cual sólo podía ser usado por una persona, donde cada parte del sistema estaba diseñada para hacer solo una cosa y hacerla

bien, de aquí el nombre; “uni” que quiere decir “mono o uno” seguido de la letra X homófona

1.2.1 Inicios

En 1990, Linus Torvals, un estudiante de 23 años de la Universidad de Helsinki, en Finlandia, comenzó a desarrollar, como hobby, un proyecto basado en el MINIX de Andrew Tenembaum. Quería llevar a cabo, sobre una computadora con procesador Intel 80386, un sistema operativo tipo UNIX que ofreciera más capacidades que MINIX. Quería aprovechar la arquitectura de 32 bits, las propiedades de conmutación de tareas que incorporaba la interfaz en modo protegido del 80386 y eliminar las barreras del direccionamiento de memoria.

Linux empezó escribiendo el núcleo del proyecto en ensamblador, y luego comenzó a añadir código en C, lo cual incrementó la velocidad de desarrollo, e hizo que empezara a tomarse en serio su idea de hacer un "MINIX mejor que MINIX". La primera versión, la 0.01 no tenía driver de disquete, y ni siquiera la dio a conocer. Llevaba incorporado un pequeño sistema de archivos y un driver de disco con muchos errores. Pero funcionaba.

En octubre de 1991, anuncio la primera versión "oficial" de LINUX, la 0.02, que ya era capaz de ejecutar el SHELL bash y el compilador gcc de GNU.

En *comp.os.minix*, un foro de discusión en Internet acerca del sistema operativo de Tenembaum, Linus Torvals escribió un llamamiento que comenzaba con una famosa frase:

¿Añoras los maravillosos días del MINIX-1.1, cuando los hombres eran hombres y escribían sus propios drivers? ¿Careces de proyectos interesantes y te mueres por desafiar a un sistema operativo que puedas modificar a tu antojo? ¿Te resulta frustrante que todo funcione con MINIX? ¿Estás harto de trasnochar para poder conseguir que funcione un programa?

Entonces, esta carta puede ser justamente para ti.

Como comenté hace un mes, estoy trabajando en una versión libre de un sistema tipo MINIX para computadoras AT-386. Finalmente ha sido mejorado el entorno, que incluso se puede utilizar, y estoy deseoso de sacar las fuentes de una distribución más potente.

Es solo la versión 0.02... pero ha conseguido que funciones bien bash, gcc, gnu-make, gnu-se, compress, etc., bajo él"

A partir de ahí, el sistema de Linus empezó a crecer. De todas partes le llegaban cartas interesándose por la idea, y comenzaron a desarrollarse proyectos

destinados a incrementar la potencia de la plataforma. Hay que mencionar que muchos de los componentes de LINUX, como drivers, protocolos o shells salieron de otro sistema UNIX de libre distribución llamado FreeBSD, desarrollado en la Universidad de Berkeley.

Una gran parte del éxito del proyecto se debió y se debe al uso de Internet, la rápida distribución de los componentes, la comunicación entre las comunidades que desarrollan los proyectos y la distribución del código y conocimiento.

Figura 1.1 Logo Linux

1.2.2 Características

Linux es fácilmente configurable y posee las características de los sistemas comerciales usados en entornos de trabajo de mucha exigencia, así, lo mismo se encuentra una instalación Linux en un servidor de un muy alto rendimiento, que en una PC de escritorio común. En gran medida todas estas posibilidades que se han ido incorporando poco a poco a Linux, se deben a la *libre* distribución de su código y las posibilidades de la licencia GNU y que esto lleva a que muchas personas, muchos usuarios, muchos grupos de usuarios, muchos desarrolladores modifiquen el código según sus propias necesidades. Esto ah hecho que las aportaciones que se le hacen poco a poco al código y, a cada distribución las haga mucho mejores versión con versión, aportación tras aportación.

Algunas de sus principales características son:

- Multiproceso. Permite la ejecución de varias aplicaciones simultáneamente.
- Multiusuario. Distintos usuarios pueden acceder a los recursos del sistema simultáneamente aunque se trate de una instalación en una sola máquina.
- Multiplataforma. Funciona con la mayoría de plataformas del mercado: Intel 386/486/Pentium, Motorola 680, Sun Sparc, etcétera.
- Shells programables que lo convierten en uno de los sistemas más flexibles que existen.

- Soporte para cualquier cantidad y tipo de dispositivos directamente en el núcleo.

- Soporte para la mayoría de sistemas de archivos.

- También podríamos citar su arquitectura modular que evita los sistemas monolíticos y permite prácticamente que el usuario se fabrique un núcleo *a la carta*. Estas características lo convierten, probablemente, en uno de los sistemas más avanzados que existen. [1]

Algo importante con respecto al proyecto GNU/LINUX es la *filosofía* con la que las personas que aportan al código del sistema lo hacen, al menos la mayoría de ellas, las que realmente se contagian de esta, hasta utópica forma de ser, como el mismo Torvals lo dio a entender cuando dijo respecto a su adhesión a la licencia GPL: "la mejor cosa que hice nunca".

Lo que al respecto dice parte de la filosofía de GNU es:

El software libre es una cuestión de libertad: la gente debería ser libre para usar el software de todas las maneras que fueran socialmente útiles. El software difiere de los objetos materiales (como las sillas, los emparedados y la gasolina) en el hecho de que puede copiarse y cambiarse mucho más fácilmente. Estas posibilidades hacen que el software sea tan útil; creemos que los usuarios de software deberían poder hacer uso de ellas.

1.2.3 Plataformas y distribuciones más comunes

Una distribución es un conjunto de aplicaciones reunidas por un grupo, empresa o persona para permitir instalar fácilmente un sistema GNU/Linux. En general se destacan por las herramientas para configuración y sistemas de paquetes de software a instalar.

La base del software incluido con cada distribución incluye el núcleo Linux, al que suelen adicionarse también varios paquetes de software. Las distribuciones incluyen, en su mayoría, herramientas y utilidades que por lo general son proyectos de código abierto o libre (GNU).

Todas las distribuciones tratan de seguir un estándar común que contribuya a una uniformidad en las mismas. Así un usuario que sepa manejar Red Hat, también sabrá manejar Suse, Mandrake, Debian, etc. Este estándar es el LSB (Linux Standard Base - www.linuxbase.org), que está desarrollado por el "Free Standards Group" y apoyado por distintas empresas. No es obligatorio seguir estas directivas

[1] *Fuente: Infosheet-Como. Autor: Ivan Casado*

para que un sistema sea considerado una distribución de Linux, pero en ese caso perderá compatibilidades con ciertos programas diseñados para Linux. [2]

Algunas de las distribuciones más comunes de Linux son:

- Debian GNU/Linux
- Gentoo Linux
- LindowsOS
- Knoppix
- Red Hat Linux
- Slackware Linux
- SuSE Linux
- Fedora
- Ubuntu

Figura 1.2 Ejemplo de distribución (Ubuntu).

Una lista mucho más completa se encuentra en la página:

www.linux.org/dist/index.html

1.2.4 Aplicaciones

Tradicionalmente, Linux ha sido un Sistema Operativo recluido al ámbito universitario y a las grandes empresas del sector de las TIC por ser Linux un sistema basado en Unix. Por ejemplo, Microsoft, en Hotmail, usa Unix y no Windows 2000.

[2] Gedda. R. (2004). Linux breaks desktop barrier in 2004: Torvalds. Retrieved January 16, 2004

Pero Linux ha madurado. Actualmente, las bondades de Linux son accesibles a cualquiera. El entorno gráfico de Linux es muy potente, muy configurable y tan manejable como el de Windows.

En el ámbito de las Aplicaciones, Linux tiene ya una amplísima.

Aplicaciones como OpenOffice o StarOffice tienen prácticamente las mismas características que Microsoft Office y el manejo de los tres es muy similar.

Pero, para la empresa, lo más interesante de Linux según los estudios es ser libre. La economía de un sistema libre como Linux (es gratis) y el hecho de evitar problemas con las licencias, hacen de Linux un Sistema Operativo muy apropiado para la empresa y de aquí el crecimiento que está demostrando Linux en su implantación en empresas.

1.2.4.1 Comandos básicos

Los siguientes comandos se agruparon en 2 formas: Comandos de sistema más comunes y comandos relacionados con la administración.

Comandos de sistema:

ls
Descripción: list. Listar contenido de directorios.
Ejemplos: ls, ls -l, ls -fl, ls --color

cp
Descripción: copy. Copiar archivos/directorios.
Ejemplos: cp -rfp directorio /tmp, cp archivo archivo_nuevo

rm
Descripción: remove. Borrar archivos/directorios.
Ejemplos: rm -f archivo, rm -rf directorio, rm -i archivo

mkdir
Descripción: make dir. Crear directorios.
Ejemplos: mkdir directorio

rmdir
Descripción: remove dir. Borrar directorios, deben estar vacios.
Ejemplos: rmdir directorio

mv
Descripción: move. Renombrar o mover archivos/directorios.
Ejemplos: mv directorio directorio, mv archivo nuevo_nombre, mv archivo a_directorio

date
Descripción: gestion de fecha de sistema, se puede ver y establecer.
Ejemplos: date, date 10091923

history
Descripción: muestra el historial de comandos introducidos por el usuario.
Ejemplos: history | more

more
Descripción: muestra el contenido de un archivo con pausas cada 25 líneas.
Ejemplos: more archivo

grep
Descripción: filtra los contenidos de un archivo.
Ejemplos: cat archivo | grep cadena

cat
Descripción: muestra todo el contenido de un archivo sin pausa alguna.
Ejemplos: cat archivo

chmod
Descripción: cambia los permisos de lectura/escritura/ejecucion de archivos/directorios.
Ejemplos: chmod +r archivo, chmod +w directorio, chmod +rw directorio -R, chmod -r archivo

chown
Descripción: change owner. cambia los permisos de usuario:grupo de archivos/directorios.
Ejemplos: chown root:root archivo, chown pello:usuarios directorio -R

tar
Descripción: Tape Archiver. Archivador de archivos.
Ejemplos: tar cvf archivo.tar directorio , tar xvf archivo.tar, tar zcvf archivo.tgz directorio, tar zxvf archivo.tgz

gunzip
Descripción: descompresor compatible con ZIP.
Ejemplos: gunzip archivo

rpm
Descripción: gestor de paquetes de redhat. Para instalar o actualizar software de sistema.
Ejemplos: rpm -i paquete.rpm, rpm -qa programa, rpm --force paquete.rpm, rpm -q --info programa

mount
Descripción: montar unidades de disco duro, diskette, cdrom.
Ejemplos: mount /dev/hda2 /mnt/lnx, mount /dev/hdb1 /mnt -t vfat

umount
Descripción: desmontar unidades.
Ejemplos: umount /dev/hda2, umount /mnt/lnx

wget
Descripción: programa para descargar archivos por http o ftp.
Ejemplos: wget http://www.rediris.es/documento.pdf

lynx
Descripción: navegador web con opciones de ftp, https.
Ejemplos: lynx www.ibercom.com, lynx --source http://www.ibercom.com/script.sh | sh

ftp
Descripción: cliente FTP.
Ejemplos: ftp ftp.ibercom.com

whois
Descripción: whois de dominios.
Ejemplos: whois ibercom.com

who
Descripción: muestra los usuarios de sistema que han iniciado una sesion.
Ejemplos: who, w, who am i

mail
Descripción: envió y lectura de correo electrónico.
Ejemplos: mail pepe@ibercom.com < archivo, mail -v pepe@ibercom.com < archivo

sort
Descripción: ordena el contenido de un archivo.
Ejemplos: cat /etc/numeros | sort, ls | sort

ln
Descripción: =link. para crear enlaces, accesos directos.
Ejemplos: ln -s /directorio enlace

tail
Descripción: muestra el final (10 líneas) de un archivo.
Ejemplos:tail -f /var/log/maillog, tail -100 /var/log/maillog | more

head
Descripción: muestra la cabecera (10 lineas) de un archivo.
Ejemplos: head archivo, head -100 /var/log/maillog | more

file
Descripción: indica de que tipo es un archivo.
Ejemplos: file archivo, file *

Comandos de administración:

sysctl
Descripción: Configurar los parámetros del kernel en tiempo de ejecución.
Ejemplos: sysctl -a

ulimit
Descripción: muestra los límites del sistema (máximo de archivos abiertos, etc..)
Ejemplos: ulimit

adduser
Descripción: agregar un usuario de sistema.
Ejemplos: adduser pepe, adduser -s /bin/false pepe

userdel
Descripción: = eliminar un usuario de sistema.
Ejemplos: userdel pepe

usermod
Descripción: = modificar la cuenta de un usuario de sistema
Ejemplos: usermod -s /bin/bash pepe

df
Descripción: = disk free. Espacio en disco disponible.
Ejemplos: df, df -h

uname
Descripción: =unix name. Información sobre el tipo de unix en el que estamos, kernel, etc.
Ejemplos: uname, uname -a

netstat
Descripción: la información sobre las conexiones de red activas.
Ejemplos: netstat, netstat -ln, netstat -l, netstat –a

ps
Descripción: =proccess toda la información sobre procesos en ejecución.
Ejemplos: ps, ps -axf, ps -A, ps -auxf

free
Descripción: muestra el estado de la memoria RAM y el SWAP.
Ejemplos: free

ping
Descripción: herramienta de red para comprobar entre otras cosas si llegamos a un host remoto.
Ejemplos: ping www.rediris.es

traceroute
Descripción: herramienta de red que nos muestra el camino que se necesita para llegar a otra máquina.
Ejemplos: traceroute www.rediris.es

du
Descripción: =disk use. uso de disco. Muestra el espacio que esta ocupado en disco.
Ejemplos: du *, du -sH /*, du -sH /etc

ifconfig
Descripción: =interface config. Configuración de interfaces de red, módems, etc.
Ejemplos: ifconfig, ifconfig eth0 ip netmask 255.255.255.0

route
Descripción: gestiona las rutas a otras redes.
Ejemplos: route, route -n

iptraf
Descripción: muestra en una aplicación de consola todo el trafico de red IP, UDP, ICMP.
Permite utilizar filtros, y es útil para diagnostico y depuración de firewalls
Ejemplos: iptraf

tcpdump
Descripción: vuelca el contenido del trafico de red.
Ejemplos: tcpdump, tcpdump -u

lsof
Descripción: muestra los archivos(librerías, conexiones) que utiliza cada proceso
Ejemplos: lsof, lsof -i, lsof | grep archivo

lsmod
Descripción: Muestra los módulos de kernel que están cargados.
Ejemplos: lsmod

modprobe
Descripción: Trata de instalar un modulo, si lo encuentra lo instala de forma temporal.
Ejemplos: modprobe ip_tables, modprobe eepro100

rmmod
Descripción: Elimina módulos del kernel que están cargados
Ejemplos: rmmod <nombre de modulo>[3]

1.2.4.2 Procesamiento de Palabras y Procesamiento de Texto

Los editores de texto más utilizados en linux son **ed**, **vi** y **emacs**.

1.2.4.3 El editor básico de líneas ed

Este es un editor de líneas. Las operaciones del editor se realizan sobre líneas, algunos de los comandos también pueden hacer referencia a un conjunto de líneas, sin embargo, no tiene la facilidad de movimiento entre líneas como el caso de los editores de pantalla completa.

El editor Ed tiene dos modos de operación:

- Modo de comando: se utiliza para dar órdenes, como escribir en un archivo, buscar cadenas de texto, etc.

- Modo de entrada: se utiliza para introducir texto, para poder retornar el modo de orden se utiliza ^d.

Para iniciar el editor se utiliza el siguiente comando: ed archivo, si existe en archivo, copia el contenido del mismo en la memoria y muestra el tamaño en caracteres. Si el archivo no existe, señala que el buffer asignado al archivo está vacío. En ambos casos el editor queda en modo comando.

El comando P hace que se muestre el prompt del editor cuando se encuentra en modo comandos. Por omisión es el carácter *.

Cuando se comete un error aparece el carácter ?. La orden H activa una presentación de mensajes de error más explícita.

La mayoría de las operaciones en modo orden operan sobre la línea actual, o bien, sobre un rango de líneas delimitado por el número de línea inicial, una , (coma) y el número de línea final. En modo comando la línea actual se denomina. (punto) y la última línea del archivo $.

[3] Mackenzie, K. (2004). Linux Torvalds Q&A. Retrieved January 19, 2004

Comandos del editor ed y la función que realizan:

[lin]a

Añadir texto a partir de la línea indicada, o en la actual si no se indica

[lin]i

Inserta texto antes de línea indicada

[lin]c

Cambiar la línea o líneas indicadas

[lin]d

Borra la línea o las líneas indicadas

[lin]p

Visualiza la línea o el rango de líneas indicadas.

[lin]n

Igual que el anterior, mostrando el número de la línea

[lin]md

Mover la línea o rango de líneas a partir de la línea d

[lin]td

Copiar la línea o rango de líneas indicado a partir de la línea d

e[fic]

Carga el fichero indicado. No actúa si el fichero actual no está grabado

E[fic]

Carga el fichero indicado sin comprobar si el fichero actual está grabado o no

f[nombre]

Fija el nombre del fichero actual

[lin]r [fic]

Lee el fichero indicado añadiendo a partir de la línea indicada o de la última si no se indica

[lin]w [fic]

Graba las líneas indicadas, todas por defecto, en el fichero especificado o en el fichero actual por defecto

q

Sale del editor, emite un error si el fichero actual no fue grabado

Q

Igual que el anterior, sin comprobar si el fichero actual fue grabado

P

Activa /Desactiva el prompt del editor

H

Activa/Desactiva los mensaje explicativos de error

u

Deshace el último comando que puede deshacerse. Por ejemplo una inserción.

!orden

Ejecuta la orden especificada llamando al intérprete de comandos sin salir del editor

+[*n*]

Avanza *n* líneas o una por defecto desde la línea actual

-[*n*]

Retrocede *n* líneas o una por defecto desde la línea actual

^d

Vuelve al modo orden

[lin]g/expr/comando

Busca la expresión indicada (expr), en el rango de líneas indicado (todas por defecto) ejecutando en cada una de ellas el comando especificado, por ejemplo: g/hola/n

[lin]G/expr/

Se posiciona una a una en todas las líneas dónde ha encontrado *expr,* en el rango de líneas especificado (todas por defecto), a la espera de una orden o <intro> para continuar

[lin]v/expr/comando

Busca las líneas que **no** contengan la expresión indicada (*expr*), en el rango de líneas indicado (todas por defecto) ejecutando en cada una de ellas el comando especificado. Por ejemplo: v/hola/n

[lin]V/expr/

Se posiciona una a una en todas las líneas donde **no** ha encontrado *expr* en el rango de líneas indicado (todas por defecto) a espera de una orden o <intro> para continuar

[lin]s/expr/sust/[g]

Busca las líneas que contienen *expr* dentro del rango indicado o en la actual por defecto, sustituyendo *expr* por *sust*. Si se incluye *g* sustituye en todo el rango y no solo la primera concurrencia en ese rango.

1.2.4.4 El editor de pantallas vi.

El editor vi es un editor de pantalla completa.

De forma similar a ed, el editor vi siempre trabaja con una copia del archivo que se edita, que él mismo se encarga de mantener en un buffer. Los cambios realizados sobre el archivo solo se graban cuando se invoca algún comando de grabación.

Para invocar el editor vi se hace mediante: vi archivo**.** Si el archivo existe lo carga en el buffer de trabajo.

Además, la orden para invocar el editor vi tiene una serie de opciones, las más importantes son:

r Cada cierto tiempo vi salva el buffer de edición en disco. De forma que si se produce cualquier fallo en el sistema, siempre se encontrará una copia reciente del archivo que se editaba.

R Indica que solo se editará el archivo para consulta. No permitiéndose realizar cambio alguno sobre la información.

w*n* Normalmente vi utiliza la pantalla completa como ventana de edición. Mediante esta opción se podrá alterar el tamaño de la ventana de edición a un número de líneas determinado por *n*.

El editor **vi** tiene tres modos de trabajo diferentes:

- **Modo comando**: Es el modo en el que queda el editor al entrar. En él cada pulsación de una tecla se interpreta como un comando, no siendo posible la introducción de texto.

- **Modo texto**: Cada pulsación de una tecla se interpreta como un carácter a introducir al texto editado. Es el modo indicado para añadir información al archivo que se está editando.

- **Modo ex**: Se podrán utilizar los comandos del intérprete ***ex***, que en muchos casos son iguales que los comandos **ed**. Para pasar a este modo se debe estar en modo comando.

Pasar de un modo a otro:

Una característica interesante de **vi** es que permite la edición múltiple de archivos. Es decir, la posibilidad de editar conjuntamente varios archivos. Esto puede hacerse de dos formas:

- Al entrar mediante línea de comandos: **vi archivo1 archivo2 archivo3**. Inicialmente se editará *archivo1* , pero podemos editar el siguiente mediante el comando **n**, es necesario escribir los cambios del archivo actual antes de editar el siguiente. Para volver a editar el primer archivo de la lista utilizar el comando **rew**.
- Estando ya trabajando con **vi** mediante el comando **e *archivo2***. Si se está editando un archivo, puede llamarse a otro archivo para que se edite conjuntamente con el actual. Para editar el archivo anterior puede utilizarse el comando **e#**.

Al entrar en vi pasa directamente al Modo Comando.

Comandos del editor vi y Función que realizan

MODO TEXTO.

Cuando estamos en modo de entrada, todo el texto que introduzcamos formará parte del archivo. Los *NEWLINE* marcan el final de línea, y habilitan la siguiente. El *BACKSPACE* suprime el carácter de la izquierda, así hasta el principio de la línea, pero no se permite ir a la línea anterior. La tecla ESC permite pasar de modo entrada a modo comando.

^u + <ESC> Cancela la última inserción realizada.

^w + <ESC> Cancela la última palabra introducida.

<ESC> Vuelve al modo comando.

MODO COMANDO.

Al entrar en vi estamos en modo comando. Si estamos en modo de entrada y queremos pasar a modo comando se pulsará la tecla ESC.

Para comenzar a ejecutar un comando primero debemos situarnos en el punto en que queremos modificar el texto.

MOVIMIENTO DEL CURSOR

<ESPACIO> , ! , l

Avanza una posición a la derecha

<BS> , ! , h

Retrocede una posición a la izquierda

^n , ! , j

Avanza una línea

^p , ! , k

Retrocede una línea

+ , <intro>

Avanza al principio de la línea siguiente

-

Retrocede al principio de la línea anterior

0 (cero)

Retrocede al principio de la línea actual

$

Avanza al final de la línea actual

[n]G

Va a la línea *n* o a la última por defecto

[n]|

Va a la columna *n* de la línea actual o a la primera por defecto

w, W

Va al principio de la palabra siguiente

b, B

Va al principio de la palabra anterior

e, E

Va al final de la palabra actual

)

Va al final del fichero, al final de la última línea del texto

H

Va a la esquina superior izquierda de la pantalla

M

Va a la línea central de la pantalla

MOVIMIENTO DE LA PANTALLA

[lin]^U

Desplaza hacia arriba media pantalla o el número de líneas indicado por *lin*

[lin]^D

Desplaza hacia abajo media pantalla o el número de líneas indicado por *lin*

^F

Avanza una pantalla completa

^B

Retrocede una pantalla completa

^R, ^L

Redibuja la pantalla

INTRODUCCIÓN DE TEXTO (PASO A MODO TEXTO)

i

Inserta a partir del carácter actual

I

Inserta a partir del principio de la línea actual

a

Añade a partir del carácter actual

A

Añade a partir del final de la línea actual

o

Abre una línea bajo la actual y pasa a añadir al principio de la misma

O

Abre una línea sobre la actual y pasa a añadir al principio de la misma

BORRADO DE TEXTO

[n]x

Borra el carácter sobre la posición actual del cursor, o bien, *n* caracteres

[n]X

Borra el carácter anterior a la posición actual del cursor, o bien, *n* caracteres

d[movimiento cursor]

Borra el texto existente entre la posición actual del cursor y la posición que se alcanzará mediante el comando de movimiento del cursor especificado.

dw

Borra hasta el principio palabra siguiente

d)

Borra hasta el final del fichero

[n]dd

Borra la línea actual, o bien, *n* líneas a partir de la actual

D

Borra desde la posición actual del cursor hasta el final de la línea actual

Nota: El texto borrado se almacena en un buffer para poder ser recuperado.

MODIFICACIÓN DE TEXTO

u

Deshace el último cambio sobre el texto

U

Deshace todos los cambios sobre la línea actual

r

Sustituye el carácter sobre el que está el cursor, por el que se pulse a continuación

R

Permite rescribir texto sobre la actual posición del cursor

s

Sustituye el carácter actual sobre el texto introducido a continuación

S

Sustituye la línea actual por el texto introducido a continuación

J

Une la línea siguiente al final de la actual

cw

Cambia la palabra actual desde la posición del cursor por un nuevo texto

c)

Cambia el texto, desde la posición actual del cursor hasta el final del fichero

CONTROL DE BUFFERS (BLOQUES)

[“letra][n]yy

Almacena en el buffer indicado por letra, o en el buffer sin nombre, las *n* líneas siguientes a la actual, o la línea actual.

[“letra] y [movim_cursor]

Almacena en el buffer indicado por letra, o en el buffer sin nombre, las líneas existentes entre la posición actual y la indicada por la orden de movim_cursor

[“letra]p

Recupera el texto del buffer letra, o del buffer sin nombre, insertándolo a partir de la línea actual

["letra]P

Recupera el texto del buffer letra, o del buffer sin nombre, insertándolo antes de la línea actual

BÚSQUEDA

/patrón

Busca el patrón indicado desde la posición actual del cursor hasta el final del fichero

?patrón

Busca el patrón indicado desde la posición actual del cursor hasta el principio del fichero

n

Repite la última búsqueda

N

Repite la última búsqueda en el sentido opuesto al establecido

f[carácter]

Busca carácter dentro de la línea actual hasta el final de la misma

F[carácter]

Busca carácter dentro de la línea actual hasta el principio de la misma

; (punto y coma)

Repite la última búsqueda de carácter

, (coma**)**

Repite la última búsqueda de carácter en sentido opuesto

ZZ

Sale al intérprete de comandos almacenando el fichero actual si es necesario

. (punto)

Repite el último comando.

MODO INTERPRETE `ex'

Todos los comandos de **ex** comienzan con el carácter: (dos puntos). Tenemos que estar en modo orden.

CARGA Y GRABACIÓN

:w[fichero]

Graba el contenido del buffer del editor en el fichero indicado, o en el fichero por defecto

:w >>[fichero]

Graba el contenido del buffer del editor añadiéndolo al contenido del fichero indicado, o en el fichero por defecto

:r[fichero]

Carga el contenido de fichero, o del fichero por defecto, añadiéndolo a partir de la línea actual

:wq

Grabar el fichero actual y salir del editor

:q

Abandona el editor volviendo al intérprete de comandos

:q!

Abandona el editor volviendo al intérprete de comandos, sin realizar comprobaciones de grabación

:x

Guarda el archivo si es necesario y sale al intérprete de comandos

EJECUCIÓN DE ÓRDENES DEL SISTEMA

:r!orden

Ejecuta la orden indicada, añadiendo la salida de la misma a partir de la línea actual

:!orden

Ejecuta la orden especificada.

RECUPERACIÓN ANTE FALLOS EN EL SISTEMA

:preserve

Almacena el fichero como si se hubiese producido un fallo en el sistema

:recover [fichero]

Recupera el fichero indicado o el actual

EDICIÓN COMPARTIDA DE FICHEROS

:n

Edita el siguiente fichero de la lista con la que se llamó al editor

:n!

Edita el siguiente fichero de la lista con la que se llamó al editor, sin realizar comprobaciones de grabación

:rew

Vuelve a editar el primer fichero de la lista con la que se llamó al editor

:rew!

Vuelve a editar el primer fichero de la lista con la que se llamó al editor, sin realizar comprobaciones de grabación

:e[fichero]

Edita el fichero especificado, o el fichero por defecto

:e![fichero]

Edita el fichero especificado, o el fichero por defecto, sin realizar comprobaciones de grabación

INFORMACIÓN DEL FICHERO ACTUAL

:args

Muestra la línea de llamada al editor en la línea de estatus

:f

Muestra el nombre del fichero editado

:f nombre

Fija el nombre del archivo por defecto

ABREVIATURAS

:abbr a b

Establece que la cadena *b* es la expansión de la abreviatura *a*, cada vez que se escriba la cadena a se expandirá el texto según *b*, ejemplo:

BÚSQUEDA Y SUSTITUCIÓN

:[lin]s/expr/sust/g

Busca el texto *expr* dentro del rango indicado, sustituyéndolo por *sust*, donde:

;

Indica que es un comando de ex

[lin]

Marca el rango, expresado en números de línea, dónde se realizará la búsqueda y posterior sustitución, según el siguiente formato: [lin_inicial**,**lin_final] y dónde cada número de línea puede sustituirse por los caracteres $ (línea final), **. (**línea actual), % (referencia a todo el fichero)

s

Es el comando de sustitución

expr

Es el texto a buscar

sust

Es el texto a cambiar

g

Indica que deben sustituirse todas las apariciones del texto *expr* dentro del rango especificado

Ejemplos**: :**%s/volverán/vendrán/g

:1,$s/volverán/vendrán/g

:3,**.**s/volverán/vendrán/g

COMANDOS DE CONFIGURACIÓN DE OPCIONES

:set autoindent

Activa el modo de indentación automática, mediante el cual cada vez que se pulsa <return> para terminar una línea, el cursor se sitúa en la línea de abajo pero no en la primera columna, sino en la columna dónde empezó a escribir en la línea anterior.

:set noautoindent

Desactiva el modo de indentación automática

:set autowrite

Activa la grabación automática en la edición compartida de ficheros, mediante la cual, al editar otro fichero de la lista se graba el actual

:set noautowrite

Desactiva la grabación automática

:set ignorecase

Activa la no distinción entre mayúsculas y minúsculas al realizar búsquedas

:set noignorecase

Desactiva la no distinción entre mayúsculas y minúsculas al realizar búsquedas, se buscará el texto que coincida exactamente con el tecleado

:set number

Activa la numeración de líneas

:set nonumber

Desactiva la numeración de líneas.

:set all

Visualiza la configuración actual

:set showmode

Muestra cualquier tipo de modo en el que se esté. Al activar esta opción en la parte inferior derecha aparecen mensajes del tipo: INPUT modo texto, COMMAND modo comando, etc.

:set noshowmode

Desactiva la opción anterior showmode

CREACIÓN DE MACROS

:map nombre comando

Crea la macro llamada nombre, que realiza las acciones que especifiquen los comandos. Esta orden mapea cierta pulsación en el teclado para convertirla en un comando. Para introducir una secuencia de escape, es necesario pulsar anteriormente ^V. Otra de la secuencia necesaria para crear macros es ^M que sustituye a <return>

:map! nombre comando

Indica que la macro operará en modo texto

:unmap nombre

Deshace la macro especificada

Ejemplos de macros:

:map ^F:!date^M Para que ^F y ^M surtan efecto antes pulsar ^

:map! LL ^[:q!^M La nueva orden LL sale sin grabar desde modo texto. Para que <esc> (^[ ) surta efecto antes pulsar ^V.

:map Q :q!^M La nueva orden Q sale sin grabar desde modo comando.

OTROS COMANDOS

:w>>fichero

Añade lo editado al fichero indicado

:>n

Corre a la derecha un tabulador el número de líneas que le indiquemos, a partir de la actual

:<n

Corre a la izquierda un tabulador el número de líneas que le indiquemos, a partir de la actual

:>>n

Corre dos tabuladores a la derecha

:<<n

Corre dos tabuladores a la izquierda

:n

Desplaza el cursor a la línea que le indicamos

:sh

Escapa al shell, aparece el prompt PS1. Para volver a continuar con el editor pulsar Ctrl d o teclear exit

OPCIONES POR DEFECTO

.exrc

Si se desea trabajar con una configuración personalizada de opciones de *vi*, se puede crear el fichero .exrc en el directorio raíz del usuario. El usuario al invocar a *vi* y existir el fichero, configurará el editor con todas las opciones y comandos que contenga. Es posible que algunos comandos de configuración no funcionen correctamente por problemas de emulación[4]

1.2.4.5 El editor emacs

El nombre emacs significa Editor MACroS, que nació como una sustitución de un editor de textos anterior llamado teco. Actualmente emacs es uno de los editores más usados y ampliamente instalados disponible en el mundo Linux. Hay versiones emacs disponibles en casi todas las plataformas informáticas conocidas en el sector, desde Linux a MS-Windows.

Una versión completa de emacs es muy grande, ocupando varios megas en disco. Es un editor funcionalmente completo, muy potente y ampliado con funciones que van más allá de la edición de textos. En algunas instalaciones se puede usar para editar archivos, mantener un calendario, trabajar con e-mail, gestionar archivos, leer noticias de la red, usarlo como calculadora, etc. De alguna forma, emacs es un entorno de trabajo que contiene un editor de textos. Una versión popular de emacs se distribuye por medio de la licencia GNU. Esta versión fue creada por Richard Stallman que es uno de los patriarcas de la Free software Foundation y del proyecto GNU.

El editor emacs no dispone de los dos modos básicos que tiene *vi*. Esto significa que todo lo que escriba se coloca en la memoria intermedia. Para dar comandos al editor, para guardar archivos, buscar textos, borrarlos, etc. debe usar otras teclas. En emacs se utiliza la tecla <ctrl> y <esc> para ejecutar los distintos comandos[5].

INICIO, SALIDA Y AYUDA

emacs

Inicia el editor sin fijar ningún fichero

[4] Coffin[1989], Greenfield[].

[5] Copyright © 2006 Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

emacs fichero

Comienza la edición con el fichero especificado

<Ctrl-x><Ctrl-c>

Sale de emacs. Si no se ha guardado el fichero, pedirá confirmación

<Ctrl-h>

Llama a la ayuda de emacs

ESC

Retrocede. Salida de cualquier sitio

Copiar, cortar y pegar.

Cuando se suprime o corta caracteres estos se guardan en una memoria intermedia. Para pegar este texto se utiliza <Ctrl-y>. Este comando también llamado comando *yank*, pega el objeto a la derecha del cursor. Para realizar la operación de cortar y pegar de un bloque, hay que seguir los siguientes pasos:

- Situar el cursor al principio del texto a marcar.
- Pulsar <Ctrl-barra espaciadora> para fijar la marca.
- Mover el cursor al final del bloque de texto. Así se crea lo que emacs llama una región.
- Suprimir el texto con <Ctrl-w>.
- Si lo que se pretende es copiar el texto antes de mover el cursor hay que volver a pulsar <Ctrl-y>. Si lo que se desea es moverlo seguir con el siguiente paso.
- Situar el cursor en el punto del texto donde se quiera copiar el mismo.
- Pulsar <Ctrl-y> para pegar el bloque por debajo de la línea donde está el cursor.

COMANDOS BÁSICOS

<Ctrl-x><Ctrl-s> Guarda la memoria intermedia actual (texto editado) en disco

<Ctrl-x><Ctrl-w> Escribe la memoria intermedia actual en disco, pregunta por un nuevo nombre de archivo

<Ctrl-x><n> Cambia el nombre del archivo de la memoria intermedia actual

<Esc><z> Borra desde la posición actual hasta el carácter tecleado

<Ctrl-x><Ctrl-f> Encuentra un archivo, lo lee en una nueva memoria intermedia creada con el nombre del archivo

<Ctrl-x><Ctrl-r> Lee el archivo dentro de la memoria intermedia actual, borrando el contenido previo

<Ctrl-x><Ctrl-i> Inserta el archivo dentro de la memoria intermedia actual en la posición del cursor

<Ctrl-f> Mueve el cursor un carácter hacia adelante

<Ctrl-b>Mueve el cursor un carácter hacia atrás

<Ctrl-a>Va al principio de la línea actual.

<Ctrl-e> Va la final de la línea actual

<Ctrl-n> Va a la línea siguiente

<Ctrl-p> Va a la línea anterior

<Esc><f> Avanza una palabra

<Esc><b> Retrocede una palabra

<Esc><Shift -**.** > Va al principio de la memoria intermedia

<Esc><Shift -**,** > Va al final de la memoria intermedia

<Ctrl-d> Suprime el carácter donde se encuentra

<Ctrl-c> Inserta un espacio

<Esc><d> Suprime la palabra siguiente

<Ctrl-k> Suprime hasta el final de la línea actual. Si el cursor está al principio de la misma, la suprime entera

<intro> Inserta una nueva línea

<Ctrl-j> Inserta una nueva línea y sangra.

<Ctrl-o> Abre una nueva línea (no está claro cuando funciona, a veces funciona como un intro y otras como se indica)

<Ctrl-w> Suprime la región entre la marca y el cursor

<Esc><w>

Copia la región a la memoria intermedia de eliminación

<Ctrl-x><Ctrl-o>

Suprime la línea que está a los lados del cursor

<Ctrl-s>

Busca hacia adelante desde la posición actual.

<Ctrl-r>

Busca hacia atrás desde la posición actual

<Ctrl-s>

Repite búsqueda hacia delante

<Ctrl-r>

Repite búsqueda hacia atrás.

<Esc><Ctrl-r>

Pregunta antes de realizar una sustitución, se responde con:

<Ctrl-g> Cancelar la operación

<!> Sustituir el resto

<?> Obtener una lista de opciones

< **.** > Sustituir y salir donde se inició el comando

<y> Sustituir y continuar con la operación de sustituir

<n> No sustituir, pero continuar con la operación

<Ctrl-barra espaciadora>

Fija una marca en la posición actual del cursor

<Ctrl-x><Ctrl-x>

Intercambia la marca y el cursor

<Ctrl-w>

Suprime la región marcada

<Esc-w>

Copia la región marcada en la memoria intermedia de eliminación

<Ctrl-y>

Inserta la memoria intermedia de eliminación en la posición actual del cursor

<Ctrl-x><b>

Conmuta a otra memoria intermedia

<Ctrl-x><x>

Conmuta a la próxima memoria intermedia en la lista de memorias intermedias

<Esc><Ctrl-n>

Va al final del fichero

<Ctrl-x><k>

Suprime una memoria intermedia que no se muestra

<Ctrl-x><o>

Cuando tenemos la pantalla dividida, cambia de una a otra

<Ctrl-x><n>

Divide la pantalla en n ventanas. Ventanas con las que podemos trabajar máximo 2. Si ponemos un 3 la ventana actual se divide en dos verticalmente y podemos ir haciendo más subdivisiones.

1.2.4.6 Lenguajes y Utilerías de Programación

Los lenguajes de programación disponibles para Linux son demasiados como para mencionarlos sin que exista la posibilidad de omitir alguno.

Entre los que de manera más común pueden ser encontrados en maquinas Linux están: C, C++, Scheme, Perl, Python, Java, Fortran, Ruby, Tcl/Tk, PHP.

Pero la lista completa, tomando en cuenta los lenguajes que se generan de manera cotidiana y las nuevas versiones que son, de forma “nativa”, de otras plataformas como Windows y que se están incorporando a Linux, es complicada de realizar.

Pero se puede decir que los lenguajes principales que funcionan para Linux son:

PERL.- Su principal uso es en scripts de administración del sistema, cgi’s para páginas Web. Tiene una potente manipulación de textos y cadenas y soporta programación orientada a objetos.

PYTHON.- Se pueden programar aplicaciones para Web y para administración de sistemas, permite la programación orientada a objetos.

TCL.- Admite la programación orientada a objetos, sirve para programar aplicaciones y para administrar el sistema.

PHP.- Permite la programación orientada a objetos, a partir de la versión 5, para versiones anteriores se puede emular, muy usado para interactuar con bases de datos y programación Web, aunque sirve también para crear otro tipo de aplicaciones.

JAVA.- Permite la programación orientada a objetos, y el tipo de aplicaciones que se pueden crear con este lenguaje es muy diverso, plataformas cruzadas, www, etcétera.

LISP.- Admite la programación orientada a objetos y programación de forma funcional.

FORTRAN.- No admite programación orientada a objetos y es ideal para aplicaciones que involucran cálculos matemáticos.

Lenguaje C.- Este lenguaje es imprescindible para cualquier programador, se usa para programación de sistemas y en muy diversas aplicaciones.

C++.- Admite la programación orientada a objetos, y las aplicaciones que se pueden hacer a partir de este lenguaje son muy diversas.

También es importarte mencionar la programación del Shell, que son entornos en de programación muy importantes Linux, el dominio de este tipo de programación es de vital importancia para los administradores de sistemas.

Entre los más populares están los shell bash, tcsh, csh, ksh y zsh.

1.3 Requerimientos de Instalación

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

1.3.1 Hardware

Plataformas y Distribuciones Más Comunes

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

Las plataformas principales son:

•I386 Para equipos tipo PC compatibles.

•PPC Para equipos Maquintoch •Sparc Para estaciones de trabajo

•Alpha Para arquitectura de trabajo con procesadores en paralelo de tipo alpha.

Existen muchas distribuciones que pueden ser adquiridas por medio de Internet, haciendo la compra en el sitio oficial que le distribuye o descargándolo gratuitamente de los sitios de FTP, todas las distribución utilizan la versión del kernel que se actualiza constantemente, la diferencia radica en que los diseñadores y programadores le dan tu toque especial, principalmente en presentación de la interfaz gráfica, implementación de nuevas herramientas, programas, formatos gráficos, etc [6].

Las distribuciones más comunes son:

http://www.slackware.com/

http://www.redhat.com/

[6] Glyn Moody: *Rebel Code: Linux and the Open Source Revolution*

http://www.debian.org/

http://www.freebsd.org/

http://www.suse.com/

http://www.linux-mandrake.com/

http://www.redhat.com/

http://www.redhat.com/

http://www.redhat.com/

1.3.1.1 Motherboard y CPU

En la actualidad Linux soporta sistemas Intel 80386, 80486 o Pentium, además de ALPHA, SPARC, MIPS, PCC, AMD y Cyrix.

1.3.1.2 Requerimientos de memoria

Linux puede funcionar de forma eficiente a partir de 4 megabytes de memoria RAM, pero claro, esto depende de las necesidades del usuario, pudiendo gestionar más de 64Gb de memoria.

Linux reserva un espacio de disco duro para usarla como memoria RAM virtual: SWAP. El área SWAP no puede compararse en términos de velocidad a una memoria RAM, pero permite al sistema ejecutar aplicaciones grandes guardando en el disco duro partes de código que estén inactivas. La cantidad de espacio de intercambio depende de diversos factores; se puede aproximar un cálculo en función de la memoria física RAM.

1.3.1.3 Requerimientos de disco

La cantidad de espacio depende en gran medida de las necesidades y de la cantidad de software que se requiera instalar. El sistema base funcionará con poco menos de 100 Megabytes.

En un caso promedio, con entorno gráfico y un número aceptable de aplicaciones se puede utilizar un espacio de unos 500 Megas.

Una instalación completa ocupa alrededor de 1.2 Gb.

El núcleo soporta controladoras XT estándar, las controladoras MFM, RLL, IDE, EIDE y SCSI.

1.3.1.4 Monitor y adaptador de video.

Para trabajar en modo texto, Linux soporta todas las tarjetas de vídeo estándar Hercules, CGA, EGA, VGA, IBM monocromo y Super VGA. Los entornos gráficos como el Sistema X Window tienen requisitos propios de hardware para la tarjeta de vídeo. Se soportan prácticamente todas las tarjetas actuales, varias aceleradoras 3D y tarjetas AGP.

1.3.1.5 Interfaces de red.

Linux soporta un buen número de tarjetas Ethernet y adaptadores para LAN.

1.4 Comandos y utilerías básicas

Comandos Básicos

Estructura básica de los comandos.

comando -opciones argumento1 argumento2 ...

Siempre al inicio se teclea el comando que se quiere ejecutar, a continuación las opciones si necesitamos modificar la salida de la ejecución del comando y al final los argumentos que indican la parte donde se deposita la salida del comando.

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

Los argumentos son requisito para la ejecución del comando, si no damos el argumento apropiado para la ejecución del comando, éste no se ejecutará y mandará un mensaje de error el sistema.

uname

uname: Nos indica el tipo de LINUX que se esta utilizando.

Linux

-n Indica el nombre del servidor

atena

–r Muestra la versión del kernel

2.4.18

–m Indica la arquitectura del procesador

i586

-a Muestra todos los datos antes mencionados

Pwd

(Print Working Directory) éste comando imprime en pantalla el directorio actual de trabajo, es decir, donde nos encontramos posicionados dentro del sistema de directorios indicándonos la ruta absoluta.

Touch

Este comando nos permite crear archivos vacíos, de tamaño 0.

touch prueba

Este comando es útil cuando nos interesa generar archivos que tienen una función especial como las bitácoras.

Caracteres Especiales.

UNIX maneja varios caracteres especiales, cada uno de ellos realiza una función en específico, en ocasiones pueden ser combinados para aumentar el potencial del comando en ejecución, se recomienda que estos caracteres no se utilicen al nombrar archivos o directorios. Estos caracteres son: ;

, < > | >> ^ $ & / ~ \ [ ] ‘ “ ( ) . * ?

rmdir (remove directory)

Éste comando borra directorios, la única condición que debe cumplir es que se encuentre vacío el directorio a ser borrado.

rmdir nom_directorio

rmdir examen

Comodines *, ?

Existen dos caracteres llamados comodines los cuales pueden sustituir un caracter o conjunto de caracteres durante la ejecución de comandos, Ej.

ls ??a Muestra aquellos archivos y directorios los cuales tengan el nombre de tres letras y la tercera sea la letra a

ls *u*t* Muestra aquellos archivos y directorios los cuales tengan en el nombre una r una t en cualquier posición pero en el orden que se le indica.

(Remove)

El comando rm borra archivos, puede ser borrado un solo archivo o varios haciendo uso de los meta caracteres * ?

rm saludo

rm lo*

rm pr??ba

mv (move)

El comando mv tiene dos funciones básicas, la primera es mover archivos o estructuras de directorios de un lugar a otro, la segunda es renombrar archivos o directorios.

mv saludo musica/

Aquí se está moviendo el archivo saludo dentro del directorio musica.

mv saludo hola

Aquí se está cambiando el nombre del archivo saludo, ahora se llamará hola.

1.4.1 Conceptos básicos.

Características De LINUX

•Es un sistema operativo de tiempo compartido

•Multiusuario

•Multitarea

•Multiproceso .

Filosofía De LINUX

• Su forma modular lleva a su filosofía principal “Entre más pequeño mejor”.

• Permite la reducción de esfuerzo combinando las herramientas.

• Es un sistema personalizable de a cuerdo a las necesidades de cada usuario.

Componentes Del Sistema Operativo LINUX

LINUX está compuesto básicamente de cuatro capas, la capa mas interna está conformada por el hardware, que es el conjunto de piezas físicas del equipo de cómputo.

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

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

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

1.4.1.1 Creación de una cuenta.

Esto es necesario, porque no es buena idea usar la cuenta de root para los usos “normales”. La cuenta de root debería reservarse para el uso de comandos privilegiados y para el mantenimiento del sistema.

Para crear una cuenta de usuario se necesita entrar en al sistema como root y usar las órdenes useradd o adduser.

1.4.1.2 Consolas virtuales.

La consola del sistema es el monitor y teclado conectado directamente al sistema. Linux proporciona acceso a consolas virtuales (o VC), las cuales permitirán tener más de una sesión de trabajo activa desde la consola a la vez.

1.4.1.3 Cambio de contraseña, passwd.

En *Linux*, las **contraseñas** deben de tener un mínimo de 8 caracteres y combinar letras minúsculas, letras mayúsculas y dígitos.

Para poder cambiar una contraseña en Linux el usuario root debe ejecutar el comando *passwd usuario* y el sistema de pedirá la nueva contraseña de usuario y su confirmación.

1.4.2 Primeros pasos.

Absolutamente todas las actividades de administración se planean.

1.4.2.1 Archivos y directorios.

Antes de que se puedan usar las particiones de Linux para almacenar ficheros, hay que crear los sistemas de archivos en ellas. La creación de un sistema de archivos es análoga a formatear una partición en Windows u otros sistemas operativos.
El tipo de sistema de archivos más usado es el Sistema de Ficheros Extendido 2, o ext2fs. El ext2fs es uno de los sistemas más eficientes y flexibles; permite hasta 256 caracteres en los nombres de los ficheros y tamaños de estos de hasta 4 Terabytes.

El propio proceso de instalación crea los sistemas de archivos de forma automática, sin embargo en cada instalación se puede personalizar el tamaño y numero de las particiones y directorios que se usen en el sistema de archivos del sistema.

1.4.2.2 Directorio raíz.

Directorio Raíz: En el directorio raíz se encuentran la siguiente estructura:

bin:	Binarios de comandos esenciales.
boot:	Archivos estáticos de cargador de arranque (boot.-loader).
dev:	Archivos de dispositivos.
etc:	Configuración del sistema.
home:	Directorios de usuarios.
lib:	Librerías compartidas.
mnt:	Punto de montaje de particiones temporales.
root:	Directorio home para el usuario root.
sbin:	Binarios del sistema esenciales.
tmp:	Archivos temporales.
usr:	Segunda jerarquía mayor de archivos.
var:	Información variable.

1.4.2.3 Directorio de trabajo, pwd.

El comando pwd (*print working directory*), muestra en pantalla el directorio en el que se encuentra actualmente.

1.4.2.4 Rutas relativas y absolutas.

Las rutas relativas: Son aquellas que se construyen a partir del directorio en el que nos encontramos.

Las rutas absolutas: Son aquellas que se construyen poniendo toda la ruta hasta el directorio que nos queremos mover.

1.4.2.5 Cambio de directorio, cd.

Para ir de un directorio a otro en Linux, es importante saber cuál es el directorio actual y a qué directorio se quiere ir exactamente.

Para ir directamente al directorio HOME de usuario, se puede escribir el comando cd sin ningún argumento.

Para ser trasladado a otro directorio se pueden usar rutas absolutas o rutas relativas. Las rutas absolutas se deberán de indicar desde la raíz del sistema de archivos hasta llegar al directorio solicitado. Las rutas relativas hacen referencia a donde se desea llegar, pero situando la referencia al directorio actual.

El comando cd funciona con las siguientes opciones:

cd	regresa al directorio HOME del usuario.
cd ~	regresa al directorio HOME del usuario.
cd /	lleva al directorio raíz del sistema de archivos.
cd ..	traslada al directorio superior.
cd -	lleva al directorio anterior.

1.4.2.6 Listado de directorio ls.

Por si solo ls muestra los archivos del directorio.

Existen muchos modificadores posibles, las más comunes son:

–a	muestra todos los archivos de un directorio, incluyendo los ocultos.
-l	muestra los detalles sobre el contenido, incluyendo los permisos, el propietario, el grupo, el tamaño, la fecha de creación si el archivos es o no un enlace a otro archivo o directorio de sistema y cuál es.
-r	muestra el contenido del directorio desde e final hasta el inicio (reverse).
-R	La opción recursiva lista los contenidos de todos los directorios que estén por debajo del directorio actual.
-s	clasifica los archivos por su tamaño.

1.4.2.7 Creación de directorios, mkdir.

mkdir crea directorios con los nombres especificados.

De forma predeterminada, los permisos de los directorios creados son 0777 (‘a+rwx’) menos los bits puestos 1 en la unmask

Opciones:

-m modo, --mode=modo

Establece los permisos de los directorios creados a modo, que puede ser simbólico como en chmod y entonces emplea el predeterminado como el punto de partida.

-p, --parents

Crea los directorios padre que falten para cada argumento directorio. Los permisos para los directorios padre se ponen a la unmask modificada por ‘u+rwx’. No hace caso de argumentos que correspondan a directorios existentes. Así, si existe un directorios /a, entonces ‘mkdir /a’ es una error, pero ‘mkdir –p /a’ no lo es.

1.4.2.8 Copiar archivos y directorios, cp.

cp Copia archivos y directorios. Si el último argumento se refiere a un directorio existente, cp copia cada archivo fuente a ese directorio manteniendo el mismo nombre. En otro caso, si se le dan dos archivos copia el primero sobre el segundo.
Opciones:

-a, --archive
Reserva tanto como sea posible la estructura y atributos de los archivos originales en la copia

-d, --no-dereference

Copia los enlaces simbólicos como tales en lugar de copiar los archivos a los que apunten, y preserva las relaciones en los enlaces duros entre archivos fuente en las copias.

-f, --force

Elimina los archivos de destino que ya existen sin pedir confirmación.

-i, --interactive

Pregunta si sobrescribir archivos de destino regulares existentes

-l, --link

En vez de hacer copias de archivos que no son directorios, hace enlaces duros.

-p, --preserve
Preserva los permisos, el propietario, el grupo y los tiempos de los archivos originales.

-P, --parents

Forma el nombre de cada archivo de destino añadiendo al directorio destino una barra inclinada y el nombre especificado del archivo origen. El último argumento dado a cp debe ser el nombre de una directorio existente.

-r

Copia directorios recursivamente

-R, --recursive

Copia directorios recursivamente, preservando los no-directorios.

-s, --symbolic-link

Hace enlaces simbólicos en vez de copias de archivos que no sean directorios.

-u, --update

No copia un archivo no-directorio si el destino ya existe y tiene el mismo tiempo de modificación o más reciente.

-v, --verbose

Muestra el nombre de cada archivo antes de copiarlo.

-x, --one-file-system

Se salta subdirectorios que están en sistemas de archivos diferentes de aquél en el que empezó la copia.

1.4.2.9 Mover o renovar archivos y directorios, mv.

mv mueve o renombra archivos o directorios.

Si el último argumento nombra a un directorio existente, **mv** mueve cada uno de los otros ficheros a un fichero con el mismo nombre en ese directorio. Si no, si se le se dan dos ficheros, renombra el primero al segundo. Es un error que el último argumento no sea un directorio y se den más de dos archivos.

1.4.2.10 Borrar archivos y directorios rm, rmdir.

rm borra cada *fichero* dado. Por lo normal, no borra directorios. Pero cuando se da la opción -r o -R, se borra el árbol de directorios entero a partir del directorio especificado (y sin limitaciones en cuanto a la profundidad de los árboles de directorio que pueden borrarse con `rm -r').

Si se da la opción -i, o si un fichero no es modificable, y la entrada estándar es una terminal, y la opción **-f** no se ha dado, **rm** pregunta al usuario si quiere borrar realmente el archivo, escribiendo una pregunta en la salida estándar de errores y leyendo una respuesta desde la entrada estándar. Si la respuesta no es afirmativa, el archivo no se borra y se pasa al siguiente.

1.4.2.11 Mostrar contenido de archivos, cat, more.

cat concatena un archivo y muestra el resultado en la salida estándar.

more es un filtro para paginar a través de un texto una pantalla a la vez.

1.4.2.12 Ayuda en línea, man.

man formatea y muestra las páginas del manual en línea

1.5 Atributos de archivos.

En el sistema de archivos de Linux existen ciertos atributos que pueden ayudar la incrementar la seguridad de un sistema.

A Don’t update Atime
S Synchronous updates
a Append only
c Compressed file
i Inmutable file
d No dump
s Secure deletion
u Undeletable file

1.6 Ligas

Ligas

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

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

1.6.1 Ligas duras y suaves.

Hay en Unix dos conceptos de `enlace', llamados usualmente enlace duro (o físico) y enlace blando (o simbólico). Un enlace duro es simplemente un nombre para un archivo. (Y un archivo puede tener varios nombres. Se borra del disco solamente cuando se elimine el último nombre. El número de nombres lo muestra **ls**. No existe el concepto de nombre `original': todos tienen la misma categoría. Usualmente, pero no necesariamente, todos los nombres de un archivo se encuentran en el sistema de archivos que también contiene sus datos.)

Un enlace blando (o enlace simbólico, o acceso directo) es algo completamente diferente: es un archivo especial que contiene un nombre de camino. Así, los enlaces blandos pueden apuntar a archivos en sistemas de archivos diferentes (posiblemente montados por NFS desde máquinas diferentes), y no tienen por que apuntar a archivos que existan realmente. Cuando se accede a ellos (con las llamadas al sistema **open** o **stat**), el núcleo del sistema operativo reemplaza una referencia a un enlace blando con una referencia al archivo nombrado en el nombre de camino. (Sin embargo, con **rm** y **unlink** se borra el mismo enlace, no el fichero al cual apunte. Existen las llamadas al sistema especiales **lstat** y **readlink** que leen el estado de un enlace blando y el nombre de archivo al cual apunte. Para algunas otras llamadas al sistema, entre distintos sistemas operativos hay algunas variaciones y faltas de certeza sobre si la operación actúa en el mismo enlace, o sobre el archivo al que apunte.)

ln crea enlaces entre archivos. De forma predeterminada, hace enlaces duros; con la opción **-s**, hace enlaces simbólicos (o `blandos').

Si sólo se da un archivo, lo enlaza en el directorio en curso; esto es, crea un enlace a ese fichero en el directorio de trabajo, con el nombre igual al (último componente) del archivo. (Esto es una extensión de GNU.) De otro modo, si el último argumento se refiere a un directorio existente, **ln** crearán enlaces a cada archivo *origen* mencionado, en ese directorio, con un nombre igual al (último componente) de ese archivo *origen*. De otra forma, si sólo se dan dos archivos,

crea un enlace llamado *dest* al fichero *origen*. Es un error que el último argumento no sea un directorio y que se den más de dos archivos.

De forma predeterminada, **ln** no borra ficheros o enlaces simbólicos que ya existan. (Así, puede emplearse para propósitos de bloqueo: tendrá éxito solamente si *dest* no existe ya.) Pero se le puede forzar a borrarlos con la opción -f.

En implementaciones existentes, si es que fuera posible hacer un enlace duro a un directorio, esto sólo debe ser hecho por el súper-usuario. POSIX prohíbe que la llamada al sistema **link** y la utilidad **ln** hagan enlaces duros a directorios (pero no prohíbe que los enlaces duros crucen los límites de los sistemas de archivos).

1.7 Control de trabajos

Trabajo con archivos

cat nos permite ver el contenido de uno o mas archivos

cat archivo1 archivo2 ...

–n Utilizando la opción el comando imprime el contenido del o los archivos numerando las líneas en forma ascendente.

cat -n archivo1 archivo2 ...

more

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

more archivo1 archivo2 ...

head

nos muestra las primeras 10 líneas contenidas en un archivo

head nom_archivo

-n esta opción le indica al comando que despliegue las primeras n número de líneas en lugar de 10.

head –20 nom_archivo

tail

nos muestra las últimas 10 líneas contenidas en un archivo

tail nom_archivo

-n Esta opción le indica al comando que despliegue las últimas n número de líneas en lugar de 10.

tail –20 nom_archivo

pico

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

Permite el desplazamiento del cursor mediante las teclas de navegación. Ctrl+o permite guardar el archivo, si ya tiene nombre el archivo lo guarda automáticamente, si el archivo no tiene nombre nos va a pedir el nombre con el cual lo deseamos guardar.

Ctrl+x Permite salir del editor automáticamente si no se han hecho modificaciones al archivo.

En caso de haber hecho modificaciones pide confirmación para guardar el archivo, si se desea guardar el archivo nos permite la opción de guardarlo con otro nombre.

Ctrl+c nos indica en que línea del archivo se encuentra posicionado el cursor

Ctrl+y Permite avanzar una página dentro del archivo

Ctrl+v Permite regresar una página dentro del archivo

Ctrl+w permite hacer una búsqueda de cadenas de caracteres dentro del archivo

Ctrl+k corta la línea donde se encuentra posicionado el cursor actualmente, la línea cortada la envía a un buffer de memoria, es posible cortar más de una línea para después pegarlas en otra parte del documento.

Ctrl+u este comando nos permite pegar las líneas que fueron cortadas con anterioridad.

Ctrl+r Este comando nos permite insertar el contenido de otro archivo al final del archivo que se está editando.

Ctrl+w y Crtl+t Permite despasarnos a una línea en especifico por medio de la numeración de líneas.

1.7.1 Trabajos y procesos.

El encargado de asignar una parte de la memoria a un proceso es el KERNEL de Linux (parte central del Sistema Operativo), quien decide cuánta memoria dar a cada proceso y cómo repartir la capacidad del microprocesador entre los procesos que se están ejecutando. Unos procesos serán más importantes que otros, y de alguna manera tendrán "preferencia" a la hora de pedir cálculos o instrucciones al microprocesador sobre otros procesos. Este concepto se llama PRIORIDAD de un proceso. Por lo tanto, debe quedar claro que es el kernel quien "manda" sobre los procesos y controla su ejecución simultánea y el reparto de los recursos de la máquina.

Como todo en los sistemas UNIX, y por extensión en Linux, los procesos también tienen un dueño. Ese dueño es el usuario que ejecuta el programa que da lugar al proceso. Esto viene a explicar parte de la seguridad de Linux en cuanto a permisos. Como los archivos (y dispositivos como más adelante veremos) tienen un usuario, un grupo y unos permisos determinados asignados, y también los procesos, un proceso contará de cara al sistema de archivos con los permisos que tenga el usuario que lo creó. Resumidamente, un proceso, respecto al sistema de archivos, podrá hacer tanto como el usuario al que pertenece. Un programa ejecutado por un usuario no privilegiado nunca podrá leer en `/root` o borrar o escribir cosas en `/usr/bin`. Un usuario normal no podrá tampoco modificar parámetros de procesos que no le pertenezcan. `root` podrá ejercer cualquier acción sobre cualquier proceso.

Un proceso puede crear a su vez nuevos procesos. En la práctica, todos los procesos son hijos del primer proceso ejecutado durante el arranque, el proceso "init". Así, un proceso hijo, a pesar de ser uno nuevo, guarda una relación con el proceso que lo creó (proceso padre) [7].

1.7.2 Primer plano y segundo plano.

El control de tareas es una interesante característica que bash nos ofrece. Así, cada programa o tubería que se ejecute tiene asignado un número de tarea por parte de bash (diferente al PID).

Los programas que podemos ejecutar desde una línea de comandos pueden estar en primer o segundo plano, y de hecho pueden pasar de un estado a otro. Una consola concreta está bloqueada cuando un proceso está en primer plano,

[7] Almesberger, Werner. LILO: Generic Boot Loader for Linux. Disponible electrónicamente: tsx-11.mit.edu. 3 de Julio de 1993.

mientras que si está en segundo plano la consola está libre y podemos teclear comandos en ella.

Esto está más relacionado con la terminal en sí (bash en nuestro caso) que con el concepto de proceso que hemos aprendido. Normalmente al ejecutar un comando no recuperamos el prompt hasta que éste no termina. La necesidad que nos surge es poder usar esa terminal sin abrir otra nueva a la vez que nuestro comando sigue haciendo lo que tenga que hacer.

1.7.3 Interrupción de trabajos y envío a segundo plano.

En bash, ponemos una tarea en segundo plano añadiendo un `&` al final del comando. El comando se ejecuta mientras la terminal queda libre.

```
$ sleep 10 &
[1] 6190
$ # tras 10 segundos presionamos INTRO
[1]+  Done                    sleep 10
```

El comando **sleep** simplemente espera el número de segundos del primer argumento. Al ejecutarlo en segundo plano, bash imprime la línea `[1] 6190` y nos devuelve al prompt, aunque el comando se sigue ejecutando ahora en segundo plano. El número entre corchetes es el NÚMERO DE TAREA (o, también llamado trabajo o *job*) que bash le asignó al pasarlo a segundo plano, y el número que hay a continuación (6190) es el número de proceso que ha generado la ejecución del comando. Cada vez que al presionar INTRO hay novedades en la gestión de tareas, bash nos informa de ello. En este caso, ha terminado la tarea [1].

Teniendo una tarea en segundo plano, arrojará su salida a la terminal actual a pesar de todo. Podemos usar las redirecciones que ya conocemos para evitarlo.

Para recuperar una tarea en segundo plano a primer plano, podemos usar **fg**, seguido de %N , donde N es el número de tarea que queremos recuperar, por ejemplo **fg %2** traería a primer plano la segunda tarea.

Con una tarea en primer plano, podemos usar `Ctrl+Z` y la detendremos. Ahora podemos elegir entre continuar con esta tarea en primer plano, o mandarla a segundo plano. Para continuarla en primer plano, teclearíamos **fg %1**. Para seguirla en segundo plano, usaremos **bg %1**, suponiendo que es la tarea número 1.

Un comando ejecutándose en primer plano puede ser cancelado con `Ctrl+C`, lo que a su vez hará desaparecer al proceso resultante de la tabla de procesos del kernel

1.8 Nociones de Administración

El administrador de sistemas es la persona responsable de **configurar**, **mantener** y **actualizar** el sistema o conjunto de sistemas que forman una red.

1.8.1 Iniciar y detener el sistema.

Existen varios "Boot Loaders" en Linux, muchos de ellos sirven para arrancar varios tipos diferentes de SOs y otros sirven para poder arrancar Linux desde otras plataformas. Hoy en día tienen relevancia sólo unos pocos:

- LILO: LInux LOader, el más conocido y usado.
- GRUB: GRand Unified Bootloader, un "Boot Loader" muy potente y flexible que está ganando terreno.
- Loadlin: muy utilizado para poder cargar Linux desde DOS. Incluido en la práctica totalidad de las distribuciones.
- SYSLinux: "Boot Loader" rápido y eficiente, utilizado mayoritariamente por distribuciones de diskette o CDs de instalación.
- Poof: permite arrancar Linux desde iMac's, facilitando la expansión del software libre en entornos Mac[8].

1.8.1.1 Uso de LILO.

LILO arranca el sistema, utiliza llamadas al BIOS para volcar el kernel de Linux de disco a memoria. De esto se deduce que el kernel del sistema deberá residir en un dispositivo accesible por el BIOS(nuestra partición raíz podrá, sin embargo, residir en un dispositivo no accesible por el BIOS, como un disco duro SCSI).

Un hecho llamativo del uso de LILO es que deberemos reinstalarlo cada vez que varíe nuestra configuración de arranque. Suele ser muy habitual modificar el fichero de configuración de LILO (/etc/lilo.conf) y no actualizar el sector de arranque ejecutando "lilo" (/sbin/lilo). El motivo de tener que reinstalarlo cada vez es que LILO no es capaz de acceder a los sistemas de ficheros en tiempo de arranque, no sabe montar discos, entonces lo que hace es convertir los datos de su fichero de configuración (/etc/lilo.conf) en direcciones absolutas dentro del disco (similar al proceso de compilación de un programa con direcciones de memoria absolutas).

A la hora de instalar LILO la pregunta típica es ¿dónde desea instalar LILO, en el MBR o en la partición raíz del sistema? La respuesta a esta pregunta está en función del resto de sistemas operativos que tengamos instalados en nuestra máquina. En líneas generales, si disponemos de otro SO con otro "Boot Loader" y queremos conservarlo, deberemos instalar LILO en la partición raíz (/dev/hda2, por ejemplo). Si sólo vamos a tener Linux en nuestro equipo o si queremos que el arranque se gestione a través de LILO, deberemos elegir instalarlo en el MBR.

[8] Bach, Maurice J. The Design of the UNIX Operating System.

1.8.1.2 Comandos shutdown, halt, reboot.

Ejemplos de uso:

shutdown –h 15
Apaga el sistema en 15 minutos después de ejecutarse y no reinicia.

shutdown –h now
Apaga el sistema en ese momento y no lo vuelve a reiniciar.

halt
Apaga el sistema en ese momento y no lo vuelve a reiniciar.

shutdown –r 5
Apaga el sistema en 5 minutos después de haberlo ejecutado y reinicia.

shutdown –r now
Apaga el sistema en ese momento y reinicia.

rebbot
Reinicia el sistema.

1.8.1.3 El archivo inittab.

inittab: Es el primer archivo que es leído al arranque del sistema, contiene especificaciones sobre que otros archivos deben de ser ejecutados, el nivel de arranque del sistema (2, 3 o 5),inicializa el daemon *inetd* (conocido como "Super Server").

1.8.2 Manejo de sistema de archivos

Sistema De Archivos

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

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

1.8.2.1 Montaje de sistemas de archivos.

Al montar una unidad de disco o partición, hay primero que identificar de que dispositivo se trata, que formato tiene y en donde queremos montar dicha unidad o partición. La mayoría de las tarjetas madre tienen capacidad para soportar hasta cuatro discos duros o unidades IDE -*CDROMS y Zip Drives.*

De este modo consideraremos lo siguiente:

- Disco duro o unidad IDE primaria maestra equivaldría a /dev/hda en GNU/Linux®
- Disco duro o unidad IDE primaria esclava equivaldría a /dev/hdb en GNU/Linux®
- Disco duro o unidad IDE secundaria maestra equivaldría a /dev/hdc en GNU/Linux®
- Disco duro o unidad IDE secundaria esclava equivaldría a /dev/hdd en GNU/Linux®
- Unidad de disco flexible de 3 ½ pulgadas a /dev/fd0 en GNU/Linux®
- Segunda unidad de disco flexible de 3 ½ pulgadas o unidad de cinta equivaldría a /dev/fd1 en GNU/Linux®

La sintaxis del comando mount es:

mount -t [tipo] /dev/[dispositivo] /punto/de/montaje/

1.8.3 Manejo de usuarios y grupos

Una cuenta de correo está basada en un login y un password, ésta se complementa con la dirección que indica el dominio del servidor en el cual se encuentra alojada dicha cuenta.

El login es el identificador del usuario dentro del servidor, dicho login es único, es decir, no puede haber dos cuentas en el servidor con el mismo login ya que éste es el nombre con que se conoce al usuario dentro del servidor.

carlos@pegaso.mascarones.unam.mx

Aquí el usuario carlos tiene cuenta en el servidor llamado pegaso que se encuentra en la dependencia mascarones que pertenece a la institución UNAM la cual se encuentra en México.

El password Es la clave de acceso al sistema, sólo debe conocerla el dueño de la cuenta ya que permite el acceso al servidor, a todos sus servicios disponibles y a la información del usuario.

Cuando el sistema nos pide la contraseña y nosotros la tecleamos, ésta no se imprime en el monitor por seguridad para evitar que sea vista por otra persona.

1.8.3.1 Añadir usuarios

El proceso para dar de alta un nuevo usuario es agregar una entrada al fichero /etc/passwd con los datos antes descritos, para el nuevo usuario, crear su directorio personal y asignarle una contraseña de acceso al sistema.

Realizar esto manualmente, ósea editar el fichero, crear el directorio (con el contenido necesario), no es algo complicado, pero es posible que olvidemos algún paso, por lo que es mucho más habitual utilizar alguna herramienta que tenga automatizado el procedimiento.

Comando adduser:

Sintaxis: adduser nombre_de_usuario

Esto actualiza el fichero /etc/passwd añadiendo la entrada para el usuario nuevo, crea el directorio personal (con los ficheros de configuración), y añade la entrada al fichero de los grupos, /etc/groups. Lo único que falta es asignarle una contraseña y la cuente estará operativa. El comando adduser es equivalente, en algunos sistemas a useradd.

Opciones:

- b directorio personal
Normalmente se usa el nombre de usuario dentro del directorio /home, pero podemos especificar el que deseemos con esta opción.

- e fecha de expiración
La fecha en que la cuenta del usuario dejará de estar activa.

- g grupo inicial
El grupo principal del usuario, si deseamos que sea distinto al que se crea por defecto.

- s shell por defecto
El shell que usará el usuario. Después de ejecutar adduser, asigne la contraseña con passwd para completar el proceso.

1.8.3.2 Borrar usuarios

Comando userdel:

Sintaxis: userdel nombre_de_usuario

Igual que adduser, modifica los archivos /etc/passwd, /etc/group y si se está usando, /etc/shadow, eliminando las entradas del usuario en cuestión.

1.8.4 Archivar y comprimir

Un sistema de archivos es una estructura de directorios que se utiliza para organizar y almacenar archivos. La administración del sistema de archivos es una de las tareas más importantes de administración del sistema, ya que los sistemas de archivos gestionan toda la información del sistema. Incluida la que utiliza el propio sistema operativo para su operación. Todos los programas, librerías, archivos del sistema y archivos de usuarios de Linux están organizados en sistemas de archivos.

1.8.4.1 Uso de tar

El comando tar es utilizado normalmente para archivar los archivos en el sistema. El concepto de archivar o «empaquetar» no es otro que guardar en un único archivo una lista de varios archivos, o el contenido de todo un directorio (o varios directorios).

El formato del comando tar es:

tar <opciones> <archivo2><archivo2>...<archivoN>

1.8.4.2 Uso de gzip, compress y bzipz

La herramienta clásica para comprimir es compress, pero por motivos de eficiencia ha sido desplazada en los últimos tiempos por gzip. Además gzip entiende perfectamente el formato de compress.

El programa gzip reduce el tamaño de los archivos dados mediante el algoritmo de compresión de Lempel-Ziv (LZ77).

Normalmente, el archivo a comprimir se reemplaza por otro con la extensión.gz, manteniéndose los mismos permisos, propietarios y tiempos de modificación. Si no se da ningún archivo, o si un nombre de archivo es «-», se lee de la entrada estándar, que se comprime y se manda el resultado a la salida estándar.

2. Instalación y Administración de LINUX

2.1 Objetivo

Se instalará el sistema operativo Linux en PC's y configurará la tarjeta de red, video y particiones del disco duro, así mismo se manejará el ambiente de usuario.

2.2 Introducción

Salvo algunos casos, la instalación de Linux está asistida por aplicaciones muy potentes. El paso más delicado seguramente sea el particionamiento del disco duro.

Nos aseguraremos en todo momento de no estropear las particiones anteriores y actuaremos con sumo cuidado. En una instalación simple usaremos 2 particiones: una para Swap (la memoria virtual de Linux) y otra para los archivos (normalmente se usa ext3 por defecto). Suele ser inteligente dividir el sistema en más particiones (de esta manera, un fallo de alimentación no estropea todo el sistema) sin perder espacio. Normalmente, se crean particiones aparte para almacenar los ficheros de root y para los ficheros de inicio.

Por lo demás, la instalación es intuitiva, y el otro paso que requiere atención es elegir las aplicaciones que instalaremos. La mayoría de distribuciones nos lo ponen fácil clasificándolas por categorías (textos, audio...) y dando una descripción de qué hace cada una. Mi consejo es instalar lo necesario, pues ya habrá tiempo de instalar más cosas. Si probaste con un Live-CD ya habrás usado algunas aplicaciones que no querrás dejar de instalar.

2.3 Perfil y Actividades del Administrador

El administrador de sistemas es la persona 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.

Tareas administrativas comunes:

- Administrar usuarios.
- Configuración de dispositivos.
- Hacer respaldos.
- Capacitar usuarios.
- Asegurar el sistema.
- Registrar los cambios del sistema.
- Asesorar a los usuarios.

2.3.1 Planear las actividades

Absolutamente todas las actividades de administración se planean.

Actividades del administrador.

- Mantenimiento de claves de usuarios.
- Instalación y mantenimiento de dispositivos.
    1. Impresoras.
    2. Discos.
    3. 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).
- Atención a usuarios.
- Monitoreo del sistema.
- Detección de fallas.
- Auditoria e implantación de la seguridad del sistema.

2.3.2 Establecer políticas de uso y de administración

El Administrador debe establecer políticas en:

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

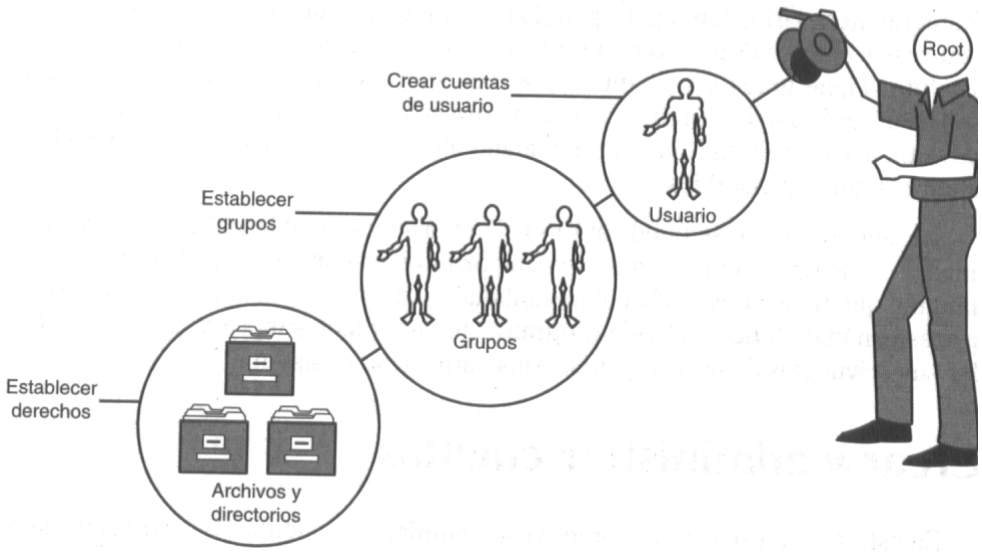


Figura 2.1 Súper usuario root.

2.4 Instalación

Hoy en día las distribuciones de Linux son distribuidas en CD-ROM e incluyen todos los paquetes completos.

Podemos omitir algunos paquetes en la instalación por ejemplo, si no se desea trabajar con X Windows podemos omitir la serie de paquetes X[1].

[1] Bach, Maurice J. The Design of the UNIX Operating System.

2.4.1Preparación de la instalación

Linux está pensado para poder trabajar en una computadora personal corriente, por tanto cualquier configuración hardware estándar será aceptada por Linux sin problemas, el problema es que Linux es de libre distribución por lo que muchos fabricantes no están dispuestos a revelar las características técnicas de los mismos a los programadores de Linux, en estas ocasiones hay que esperar a que alguien averigüe el funcionamiento del controlador correspondiente y desarrolle el controlador equivalente para Linux.

Linux funciona con cualquier procesador Intel desde el 386 SX hasta el Pentium más moderno, en el caso de que no se tenga coprocesador matemático, Linux podrá emular su funcionamiento. También funciona con procesadores clónicos como los de AMD y CYRIX. Linux posee soporte para sistema con más de un procesador Intel.

La plataforma Intel no es la única que puede soportar Linux. También ha sido llevado a otras muchas, como son los procesadores Alpha de Digital, MIPS, ARM, PowerPC, Motorola y los SPARC De Sun MicroSystems .

En lo relativo a las placas base, hay que tener en cuenta la arquitectura de bus que estas utilizan. La arquitectura de bus define la forma en que accede a los diferentes dispositivos y periféricos que están instalados. Linux en plataformas Intel puede utilizar cualquiera de los buses estándar que incorporan las placas en la actualidad, como puede ser ISA y PCI, además de otros que ya están en desuso.

Requerimientos memoria y espacio en disco:

Linux puede funcionar con un mínimo de 4 Mb de memoria RAM pero siempre será mejor cuanta más memoria RAM se tenga. El espacio requerido por Linux depende de mucho de la distribución elegida y el número de paquetes que se vaya a instalar. Una instalación completa de una distribución como Debian puede ocupar hasta 700 Mb en el disco duro.

2.4.2 Tipos de Instalación

Existen varias formas de instalación dependiendo desde donde queramos instalarla:

- Desde el disco duro, para ello tendremos primero que copiarlo desde un CD-ROM, se hace de esta forma cuando el BIOS (Basic input output system) no permite arrancar desde un CD-ROM, esto se da cuando la computadora en la que queremos instalar Linux es vieja. Para ello lo primero que tendremos que hacer será crear unos discos de arranque e iniciar la computadora con esos discos, para ello deberemos hacer una

copia de una carpeta que está en el CD-ROM D: \bootdisk con el ejecutable rawrite.exe y luego una vez arrancado se le indica la ruta desde donde debe copiar los paquetes de la instalación de Linux, que deben de estar en el disco duro.

- Otro método para poder copiar la los archivos de Linux a la computadora es el de obtener los paquetes por la red, local o por Internet, pero este último dada la extensión del mismo no es nada recomendable.
- El último método que es más usual, porque el BIOS lo soportaba es el de arrancar desde el CD-ROM teniendo en cuenta la utilidad de contar con los CD's de instalación a la mano para instalar algún paquete adicional.

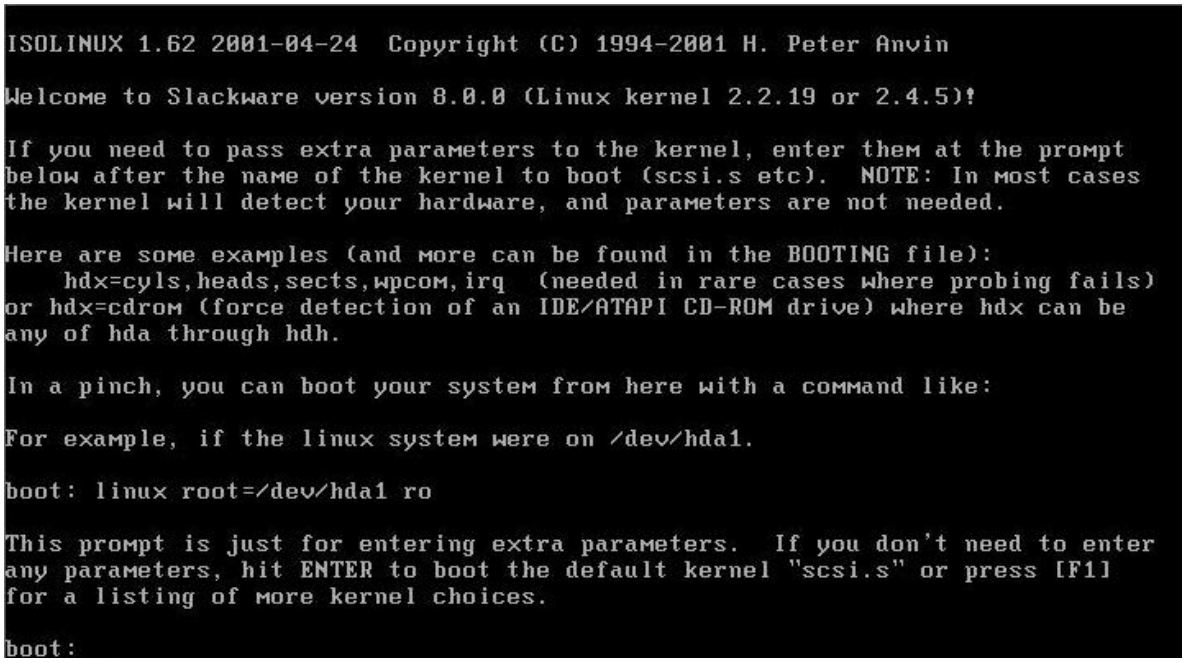


Figura 2.2 Arrancamos la instalación (vía CD ROM o disco de arranque),

2.5 Realización de Respaldos

Un sistema de respaldo debe de estar perfectamente planificado. Su planificación de forma automática no siempre es posible debido a que requieren con frecuencia intervención humana. Especialmente la copias que no caben en una sola cinta (multivolumen). Lo ideal es organizar su propio sistema de respaldo en función de las necesidades de su sistema. En un sistema es conveniente clasificar las diferentes partes de la información atendiendo a su importancia. Lo ideal sería hacer respaldo de todo regularmente, manteniendo distintas copias en rotación, es decir, si tenemos tres cintas para almacenar respaldos (Cinta A, Cinta B, y Cinta C) deberemos ir grabando siguiendo una secuencia circular A, B, C, A, B, C,... La frecuencia de los respaldos se determina en función de lo importante que sean los

datos. Generalmente es bueno hacer una copia diaria al menos de la parte del sistema que suponga mayor perdida de horas de trabajo en caso de incidente. De todas formas hay que tener en cuenta que este sistema solo nos permite mantener copia de los últimos tres o cuatro días. Esto no es suficiente porque muchas veces nos interesa recuperar una información de hace una semana, un mes, o incluso más tiempo. Por ello algunas veces se usan rotaciones de siete cintas (una para cada día de la semana) y una vez al mes se saca otra copia en una cinta distinta que se conserva por un espacio de tiempo más largo según convenga.

Una práctica común es guardar regularmente copias en otro edifico distinto para protegerse de esa forma de incendios, inundaciones, terremotos, robos. Si después de un desastre conservamos los datos vitales de nuestro negocio quizás podamos resurgir de nuestras cenizas más tarde.

Algunas veces no es posible hacer muchas copias de todo el sistema con demasiada frecuencia. Por ello se pueden delimitar áreas con distinto grado de necesidad de respaldo. El sistema operativo y las aplicaciones no suelen variar y pueden ser recuperadas desde CDs de instalación. Es importante siempre mantener copias de todo el sistema. En cualquier caso siempre ha de usarse rotación de cintas porque usar una sola copia es muy arriesgado.

Las cintas son razonablemente seguras pero se pueden romper con el uso. De todas formas es importante que la planificación de respaldo del sistema no resulte complicada. Trabajar con demasiados conjuntos de cintas distintos no es bueno.

En la copia general de todo el sistema seguramente se guardarán cosas que varían poco pero que quizás costaron mucho trabajo configurar. Por tanto es una copia que no necesita hacerse con frecuencia pero deberemos disponer siempre de algunas copias en buen estado aunque no sean muy recientes.

Todas las copias deben ser sometidas de forma automática a un proceso de verificación que en el caso más sencillo consiste en leer la lista de ficheros que contiene. No es muy fiable pero al menos detectaremos los fallos más importantes. En el mejor de los casos la comprobación se hace sobre el contenido completo comparándolo con el original. En este caso cuando la copia termina el original puede haber cambiado y por ello la detección de diferencias entre original y copia puede deberse a un cambio en el original en lugar de un error en la copia.

Una cosa que se usa mucho son las copias globales combinadas con posteriores copias incrementales. Es un sistema cuya principal ventaja es la comodidad.

Copia incremental significa que solo salvamos lo último que ha cambiado desde la última copia global. Por eso cada nueva copia incremental resulta generalmente de mayor tamaño que la anterior.

Si la copia global ocupa varias cintas. Podemos hacer copias incrementales hasta que debido a su progresivo aumento de ocupación no quepan en una sola cinta

por ejemplo, momento en el que se puede hacer una nueva copia global. En general podemos hacer una nueva copia global cuando la incremental empiece a resultar incomoda debido a su progresivo aumento de tamaño.

Este sistema de copia hace recaer demasiada responsabilidad en una sola cinta. La última copia global suele ser muy voluminosa y puede contener información que no se encuentre en ninguna otra cinta.

Hay actividades que pueden generar enormes archivos de datos y eso incrementará mucho la ocupación de nuestra siguiente copia incremental. Por ejemplo si estamos generando una imagen animada de gran tamaño o si estamos descargando de Internet imágenes de CDs enteras deberíamos guardar esos datos en un sistema de archivos especialmente dedicado a ello. De esta forma cuando hagamos respaldo podemos desmontar antes esa partición para poder realizar respaldos independientes con la frecuencia que convenga en cada caso.

Siempre que se añadan paquetes, aplicaciones, se configure algo conviene volver a sacar una nueva copia global de todo el sistema.

Para realizar copias de seguridad podemos usar aplicaciones completas pensadas para ello o fabricarnos algo a medida combinando una serie de utilidades de las cuales la mayoría ya han sido mencionadas alguna vez en capítulos anteriores. Esto último es lo que vamos a hacer nosotros porque así podrá diseñar algo personalizado y también porque resulta más didáctico.

Los programas más útiles son aquellos que sirven para empaquetar dentro de un solo archivo una gran cantidad de archivos y directorios, como por ejemplo 'tar', 'cpio', y 'afio'.

find Se usará para elaborar una lista de archivos que deseamos salvar. Su versatilidad es enorme. La opción -newer resultará de especial interés en el caso de desear hacer copias incrementales. Observe que nosotros nos situaremos previamente en un directorio adecuado y luego usaremos 'find .' para pasar a find nombres que empiezan con '.'. Es decir usaremos caminos relativos en lugar de caminos absolutos. Esto tiene su importancia porque de ese modo luego podremos recuperar la información situándola en otro lugar distinto al original si fuera necesario.

egrep Se usa para intercalarlo entre la salida de 'find' y la entrada de 'cpio' de esa forma se aumenta la flexibilidad de uso. Se puede usar para incluir determinados archivos que cumplan determinada expresión regular y/o para excluir determinados ficheros que cumplan determinada expresión regular. Para excluir se usa la opción -v.

gzip Es el compresor de uso más frecuente en Linux. Esta muy probado y es muy fiable. Cuando se coloca un compresor como gzip o bzip2 a la salida de 'cpio' o de 'tar' se pierde la posibilidad de hacer copia multivolumen porque el compresor es

el que está escribiendo en el dispositivo en lugar de hacerlo 'cpio' o 'tar'. Afortunadamente 'afio' si puede hacerlo.

bzip2 Es más eficiente en el sentido de que puede comprimir algo más pero a costa de tardar mucho más.

diff Detecta diferencias de contenido entre dos ficheros. Especialmente adecuado para ficheros de texto.

md5sum Permite asociar un código de comprobación al contenido de un fichero o dispositivo, etc.

dircomp Permite comparar dos directorios analizando su contenido recursivamente.

dirdiff Permite comparar dos directorios analizando su contenido recursivamente. Es más elaborado que el anterior.

mt Es la utilidad que se usa para controlar las cintas magnéticas. Permite operaciones de rebobinado, retension, lectura de estado, etc.

dd Sirve para copiar o leer dispositivos usando el tamaño de bloque que se desee. También permite realizar ciertas conversiones.

2.6 Interfaces Gráficas

KDE: Es un entorno de escritorio con un aspecto similar al de Windows con muchas funcionalidades incorporadas. Ideal para PCs que cuentan con un mínimo de 128 Mb de memoria RAM. Algunas características sobresalientes de KDE:

- Panel al cuál se le pueden agregar numerosos applets (pequeños programas).
- Aspecto y comportamiento altamente configurables.
- Explorador de archivos muy potente llamado Konqueror.
- Centro de Control similar al Panel de Control de Windows.

GNOME: Es la principal alternativa existente a KDE para quienes están acostumbrados a Windows. Seguramente GNOME es una buena opción para quienes prefieren un entorno menos barroco que KDE, aunque quizás posea a primera vista una interfaz menos familiar para los usuarios de Windows.

XFce: Este es un entorno de escritorio, bastante más sencillo que KDE y que GNOME, pero que es excelente para PCs que no poseen la memoria suficiente como para usar KDE o GNOME.

3. EDITORES WEB

3.1 Objetivo

Se elaborarán paginas WWW apoyado con HTML

3.2 Introducción

HTML es un lenguaje para el marcado de textos e hipertextos. Puede que esta definición sea en principio algo confusa, pero veremos que no es así. Realmente este tipo de lenguajes ha sido y sigue siendo de uso común en muchos entornos. ¿Alguna vez ha entregado un examen para su corrección por parte del profesor?, pues bien, el examen una vez corregido y devuelto está lleno de "marcas" o correcciones que le indican donde se ha equivocado y cual debería ser la respuesta correcta. Pues HTML no es algo muy distinto de esto.

HTML permite "marcar" un documento para que el navegador que lo visualice sepa como tiene que mostrarlo. Para ello, hace uso de las "marcas" o "tags" que formatearan el documento. Por ejemplo, para que el navegador mostrará un texto en negrita, habría que indicarle al navegador que ese texto ha de formatearse en negrita. Esto se conseguiría en HTML encerrando el texto en cuestión entre las marcas correspondientes.

Como el HTML va incluido en el mismo documento, junto con el texto a formatear, hay que seguir unas normas sintácticas para que el navegador sepa que debe ser interpretado como "tag" y que debe ser texto formateado con los "tags". Toda marca de HTML ha de ir encerrada entre los símbolos "< >" (menor que, mayor que). La mayoría de las marcas deben tener una marca de apertura y otra de cierre. La marca de cierre es igual a la de apertura con la excepción de que el "tag" ha de ir precedido por el símbolo "/". En nuestro ejemplo anterior, el "tag" de apertura es "" mientras que el "tag" de cierre es "". Entre el "tag" de apertura y el "tag" de cierre va incluído el texto que será formateado por el navegador para mostrarlo en negrita en este caso.

No todas los "tags" de HTML requieren la marca de cierre, aunque es bueno acostumbrarse a incluírlo en todos las marcas de HTML. Estas marcas de HTML que no requieren "tag" de cierre se denominan marcas abiertas.

3.3 Quanta

Quanta es un editor HTML que ha heredado código del Bluefish siendo lo más parecido que hay en Linux al Homesite para Windows. También se puede usar como editor de código, aunque está más pensado para editar colecciones de fuentes agrupadas en proyectos, como sería los archivos de una página Web, de la que puede sacar su estructura de imágenes, enlaces, etc.

La licencia de Quanta+ es GPL, mientras que la de Quanta Gold es privada. Sin embargo, ambos productos se parecen a tal punto que parece imposible que no compartan código[1].

3.4 Kdevelop

Con KDevelop se puede programar en C/C++ bajo un completísimo entorno de desarrollo. El proyecto KDevelop fue iniciado en 1998 para diseñar un entorno de desarrollo integrado, fácil de usar, para C/C++ en Unix. Desde entonces está disponible públicamente bajo licencia GPL, y soporta KDE/Qt, GNOME, C y C++.

Los requisitos mínimos del sistema indican que esta versión del Kdevelop sólo funciona con el KDE 2.1 o superior. Si se usa un KDE anterior se deberá usar una versión del Kdevelop anterior a ésta.

KDevelop maneja:

- Todas las herramientas de desarrollo necesarias para programar en C++, tales como el compilador, el enlazador, el automake y el autoconf.

- Al KAppWizard (asistente de aplicaciones), que genera aplicaciones de referencia, completas y listas para utilizar.

- Un generador de clases, para crear nuevas clases e integrarlas en su proyecto.

- Un explorador de ficheros para los fuentes, las cabeceras, la documentación, etc. a ser incluidos en el proyecto.

- La creación de manuales de usuario en SGML y la generación automática de los ficheros HTML con el estilo del KDE.

[1] `http://quanta.kdewebdev.org/`

- Documentación automática en HTML de las API de las clases del proyecto con referencias cruzadas a las bibliotecas utilizadas.

- Soporte para la internacionalización de su proyecto, permitiendo a los traductores añadir fácilmente su idioma al proyecto

- WYSIWYG (del inglés, lo que ve es lo que consigue) - creación de interfaces de usuario con el editor de diálogos incorporado.

- Control de su proyecto por medio de un sistema de control de versiones (CVS), proveyendo de un interfaz para la mayoría de las funciones necesarias.

- Depurado de su aplicación en conjunción con el KDbg

- Edición de los mapas de pixeles del proyecto con el programa KIconEdit.

- La posibilidad de incluir cualquier otro programa que usted considere necesario para el desarrollo de programas, mediante el menú de 'Herramientas'.

El visualizador de clases y el buscador de errores le llevan a cualquier parte del código del proyecto con un solo click, sin necesidad de buscar archivos; los árboles de archivos le dan acceso directo a los archivos del proyecto y el sistema integrado de ayuda le ofrece acceso a la documentación en línea desde cualquier parte del IDE.

4. ADMNINISTRACIÓN DE SERVIDORES WWW CON LINUX

4.1 Objetivo

Se identificará el procedimiento para instalar, configurar y administrar su propio servidor de WWW en un servidor de plataforma Linux.

4.2 Introducción

Servidor WWW es llamado a un servidor que ofrece servicios dentro del Word Wide Web en Internet, es un programa encargado de ofrecer comunicación mediante el protocolo http.

HTTP – Hypertext Transfer Protocol.

¿Qué es HTTP ?

HTTP es el protocolo de red para el 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.

Simple y Poderoso.

¿Cómo funciona HTTP ?

1.- El cliente HTTP abre una conexión.

2.- El server manda un “acknowledge” notificando que se ha abierto una sesión.

3.- El cliente envia su “request message” solicitando un recurso.

4.- El servidor responde con “response message” que contiene el recurso solicitado y cierra la conexión.

4.2.1 Diagrama de bloques del funcionamiento de un servidor WWW.

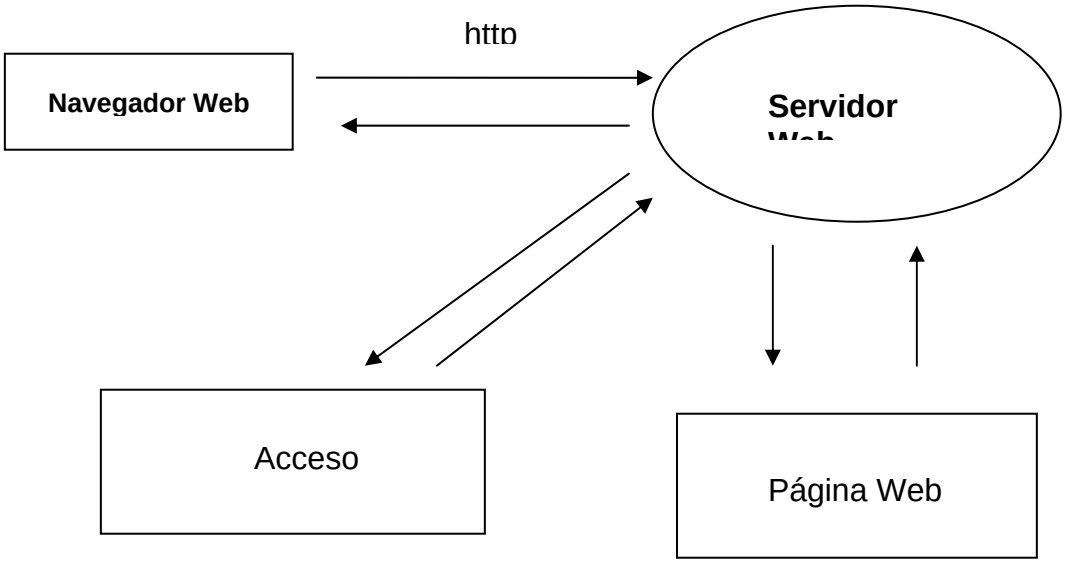


Figura 4.1 diagrama de bloques del funcionamiento de un servidor WWW.

4.3 Instalación de un servidor WWW

Se debe de conseguir el archivo fuente de Apache desde: http://httpd.apache.org

Se debe de descomprimir el archivo en un directorio temporal, por lo general /tmp, después hay que entrar en el directorio en donde se descomprimió el archivo.

Se genera un directorio con nombre: apache-<número de versión> dentro de este directorio temporal. Dentro de este directorio habrá que ejecutar el comando:

./configure –prefix=/ruta/de/instalación

Posteriormente hay que ejecutar:

make; make install

Si todo tiene éxito, la instalación queda lista.

Se puede levantar el demonio httpdd con el comando:

apachectl start

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

bin: archivos ejecutables del Apache.

conf: archivos de configuración del servidor.

error: archivos con los mensajes de error del servidor, en varios lenguajes.

htdocs: directorio raíz por defecto del servidor (Se guardan las páginas Web).

icons: directorio donde se encuentran los iconos que utiliza el servidor (entre otras cosas para mostrar estructuras de directorios).

logs: directorio donde se almacenan los registros de acceso y errores del servidor.

manual: directorio donde se encuentra el manual del Apache.

proxy: Directorio con los ficheros de la caché del servidor.

4.4 Configuración del un servidor

El servidor Apache es un software que esta estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo.

Los módulos del Apache se pueden clasificar en tres categorías:

- Módulos Base: Módulo con las funciones básicas del Apache
- Módulos Multiproceso: son los responsables de la unión con los puertos de la máquina, acepando las peticiones y enviando a los hijos a atender a las peticiones
- Módulos Adicionales: Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.

El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software[1].

Módulos Base y Módulos Multiproceso:

core: Funciones básicas del Apache que están siempre disponibles.

mpm_common: Colección de directivas que se implementan en más de un módulo multiproceso.

beos: Módulo de multiproceso optimizado para BeOS.

leader: Variable experimental de MPM.

mpm_netware: Módulo de multiproceso que implementa un servidor Web optimizado para Novell NetWare.

mpmt_os2: MPM híbrido, multiproceso y multihilo para OS/2.

perchild: Módulo multiproceso que permite a los procesos demonio servir las peticiones que se asignan a distintos id de usuario.

prefork: Implementa un servidor sin hilos.

threadpool: Variante experimental del módulo estándar de MPM.

mpm_winnt: Módulo multiproceso optimizado para Windows NT.

worker: Módulo multiproceso que implementa un híbrido multihilos y multiprocesos de servidor Web.

Módulos adicionales:

mod_access: proporciona control de acceso basándose en el nombre del host del cliente, su dirección IP u otras características de la petición del cliente.

mod_actions: este módulo se utiliza para ejecutar Scripts CGI, basándose en el tipo de medio o el método de petición.

mod_alias: proporcionado para mapear diferentes partes del sistema de ficheros del servidor en el árbol de documentos del servidor, y para redirección de URL's.

[1] *www.wikipedia.org*

mod_asis: envío de ficheros que tienen sus propias cabeceras http.

mod_auth: autentificación de usuario utilizando ficheros de texto.

mod_auth_anon: permite a usuarios anónimos acceder a áreas autentificadas.

mod_auth_dbm: proporciona autentificación utilizando ficheros DBM.

mod_auth_digest: autentificación de usuario utilizando MD5.

mod_auth_ldap: permite la utilización un directorio LDAP para almacenar la base de datos de autentificación.

mod_autoindex: muestra los contenidos de un directorio automáticamente, parecido al comando ls de Unix.

mod_cache: Cache de contenidos indexados por URI's.

mod_cern_meta: Semántica de etiquetas meta del CERN.

mod_cgi: Ejecución de Scritps CGI.

mod_cgid: ejecución de Scripts CGI utilizando un demonio CGI externo.

mod_charset_lite: para la especificación del juego de caracteres de las traducciones.

mod_deflate: comprime el contenido antes de ser enviado al cliente.

mod_dir: Proporcionado para redirecciones y para servir los ficheros de listado de directorios.

mod_disk_cache: Cache para almacenar contenidos identificados por URI.

mod_echo: Un servidor simple de hecho para ilustrar los módulos del protocolo.

mod_env: modificación del entorno que se envié a los scripts CGI y las páginas SSI.

mod_expires: Generación de las cabeceras http Expires, de acuerdo de los criterios especificados por el usuario.

mod_ext_filter: pasa el cuerpo de la respuesta a través de un programa antes de enviársela al cliente.

mod_file_cache: cachea una lista estática de ficheros en memoria.

mod_headers: personalización de las peticiones HTTP y las cabeceras de las respuestas.

mod_imap: proceso de imágenes en el lado del servidor.

mod_include: Documentos HTML generados por el servidor (Server Side Includes).

mod_info: proporciona una visión comprensiva de la configuración del servidor.

mod_isapi: Extensiones ISAPI en Apache para Windows.

mod_ldap: pool de conexiones LDAP y cacheo de resultados para la utilización de otros módulos LDAP.

mod_log_config: registro de las peticiones hechas al servidor.

mod_logio: registro del número de bytes recibidos y enviados en cada respuesta.

mod_mem_cache: Caché de contenidos identificados por URI.

mod_mime: asocia las extensiones de peticiones de los ficheros con el comportamiento del fichero (manejadores y filtros) y contenido (tipos mime, idioma, juego de caracteres y codificación).

mod_mime_magic: determina el tipo MIME de un fichero mirando unos pocos bytes del contenido.

mod_negotiation: se proporciona para la negociación del contenido.

mod_proxy: servidor HTTP/1.1 proxy/gateway.

mod_proxy_connect: extensión de mod_proxy para la gestión de las peticiones CONNECT.

mod_proxy_ftp: soporte FTP para mod_proxy.

mod_proxy_http: soporte HTTP para el módulo mod_proxy.

mod_rewrite: proporciona un motor de reescritura basado en reglas que rescribe las peticiones de URL's al vuelo.

mod_setenvif: permite la configuración de las variables de entorno basándose en las características de la petición.

mod_so: carga del código ejecutable y los módulos en al iniciar o reiniciar el servidor.

mod_speling: intenta corregir las URL mal puestas por los usuarios, ignorando las mayúsculas y permitiendo hasta una falta.

mod_ssl: criptografía avanzada utilizando los protocolos Secure Sockets Layer y Transport Layer Security.

mod_status: proporciona información en la actividad y rendimiento del servidor.

mod_suexec: permite a los scripts CGI ejecutarse con un nombre y grupo específico.

mod_unique_id: proporciona variables de entorno y un identificador único para cada petición.

mod_userdir: directorios específicos para usuarios.

mod_usertrack: registro de actividad de un usuario en el sitio.

mod_vhost_alias: Proporcionado para configurar muchos servidores virtuales dinámicamente.

El archivo httpd.conf es el archivo principal de configuración del Apache, se encuentra dentro del directorio Conf, en el directorio de instalación del Apache.

En primer lugar hay que destacar que el archivo está dividido en tres secciones, que son:

- Parámetros globales
- Directivas de Funcionamiento
- Host Virtuales

En el archivo se encuentran todos los parámetros de funcionamiento de Apache. Algunos parámetros son generales para la instalación y funcionamiento del Apache. Muchos otros de los parámetros se pueden configurar independientes para un conjunto de directorios y/o archivos. En estos casos los parámetros se encuentran ubicados dentro de secciones donde se indica el ámbito de aplicación del parámetro.

4.5 ejecución del servidor

Si el puerto especificado en la directiva Listen del archivo de configuración es el que viene por defecto, es decir, el puerto 80 (o cualquier otro puerto por debajo del

1024), entonces es necesario tener privilegios de usuario root (superusuario) para iniciar Apache, de modo que pueda establecerse una conexión a través de esos puertos privilegiados. Una vez que el servidor Apache se ha iniciado y ha completado algunas tareas preliminares, tales como abrir sus archivos log, lanzará varios procesos, procesos *hijo*, que hacen el trabajo de escuchar y atender las peticiones de los clientes. El proceso principal, `httpd` continúa ejecutándose como root, pero los procesos hijo se ejecutan con menores privilegios de usuario. Esto lo controla el Módulo de MultiProcesamiento (MPM) seleccionado.

La forma recomendada para invocar el ejecutable `httpd` es usando el script de control apachectl. Este script fija determinadas variables de entorno que son necesarias para que `httpd` funcione correctamente en el sistema operativo, y después invoca el binario `httpd`. `apachectl` pasa a httpd cualquier argumento que se le pase a través de la línea de comandos, de forma que cualquier opción de `httpd` puede ser usada también con `apachectl`. Puede editar directamente el script `apachectl` y cambiar la variable `HTTPD` variable que está al principio y que especifica la ubicación exacta en la que está el binario `httpd` y cualquier argumento de línea de comandos que quiera que esté *siempre* presente.

La primera cosa que hace `httpd` cuando es invocado es localizar y leer el archivo de configuración `httpd.conf`. El lugar en el que está ese fichero se determina al compilar, pero también es posible especificar la ubicación en la que se encuentra al iniciar el servidor Apache usando la opción de línea de comandos `-f`

```
/usr/local/apache2/bin/apachectl -f /usr/local/apache2/conf/httpd.conf
```

Si todo va bien durante el arranque, la sesión de terminal se suspenderá un momento y volverá a estar activa casi inmediatamente. Esto quiere decir que el servidor está activo y funcionando. Puede usar su navegador para conectarse al servidor y ver la pagina de prueba que hay en el directorio DocumentRoot y la copia local de esta documentación a la que se puede acceder desde esa página.

5. PROGRAMACIÓN CON PHP

5.1 Objetivo

Proporcionar los conocimientos necesarios que permitan crear aplicaciones dinámicas e interactivas para el web utilizando el lenguaje PHP.

5.2 Introducción

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…
- 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[1].

[1] Copyright © 2003-2004 por el Grupo de documentación de PHP

5.2.1 Sintaxis básica

Para interpretar un archivo, PHP simplemente interpreta el texto del archivo hasta que encuentra uno de los caracteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Conjunto de etiquetas usadas para denotar código PHP. De estas cuatro, sólo dos (<?php. . .?> y <script language="php">. . .</script>) están siempre disponibles; el resto pueden ser configuradas en el fichero de `php.ini` para ser o no aceptadas por el intérprete. Mientras que el formato corto de etiquetas (short-form tags) y el estilo ASP (ASP-style tags) pueden ser convenientes, no son portables como la versión de formato largo de etiquetas. Además, si se pretende embeber código PHP en XML o XHTML, será obligatorio el uso del formato <?php. . .?> para la compatibilidad con XML.

5.2.2 Variables

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable valido tiene que empezar con una letra o un guión (underscore), seguido de cualquier número de letras, números y guiones. Como expresión regular se podría expresar como: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

En PHP3, las variables siempre se asignan por valor. Esto significa que cuando se asigna una expresión a una variable, el valor íntegro de la expresión original se copia en la variable de destino.

PHP4 ofrece otra forma de asignar valores a las variables: *asignar por referencia*. Esto significa que la nueva variable simplemente referencia a la primera, en otras palabras, "se convierte en un alias de" o "apunta a" la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Esto también significa que no se produce una copia de valores; por tanto, la asignación ocurre más rápidamente. De cualquier forma, cualquier incremento de velocidad se notará sólo en los bucles críticos cuando se asignen grandes matrices u objetos.

Para asignar por referencia, simplemente se antepone un ampersand ‘&’ al comienzo de la variable cuyo valor se está asignando (la variable fuente).

5.2.2.1 Variables

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. Sin embargo es complicado listar estas variables debido a que dependen de la instalación del servidor y podrán ser diferentes en un servidor o en otro igual que podrán estar o no disponibles.

5.2.2.2 Variables de servidor: $_SERVER

$_SERVER es una matriz que contiene información tal como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz son creadas por el servidor Web. No existen garantías de que cada servidor vaya a proveer alguno de estos valores; puede que los servidores omitan algunos, o provean otros.

Esta es una variable 'superglobal', o global automática. Esto quiere decir que está disponible en todos los contextos a lo largo de un script.

Los posibles elementos de la matriz son:

'PHP_SELF'

El nombre de archivo del script ejecutándose actualmente, relativo a la raíz de documentos.

'argv'

Matriz de argumentos pasados al script. Cuando el script es ejecutado en la línea de comandes, ésta entrega acceso al estilo ‘C’ a los parámetros de la línea de comandos. Cuando es llamado mediante el método GET, ésta contendrá la cadena de consulta (query).

'argc'

Contiene el número de parámetros de línea de comandos pasados al script (si se ejecuta en la línea de comandos).

'GATEWAY_INTERFACE'

Qué revisión de la especificación CGI está usando el servidor; i.e. 'CGI/1.1'.

'SERVER_ADDR'

La dirección IP del servidor bajo la cual está siendo ejecutado el script actual.

'SERVER_NAME'

El nombre del servidor anfitrión bajo el que está siendo ejecutado el script actual. Si el script está corriendo en un host virtual, éste será el valor definido para tal host virtual.

'SERVER_SOFTWARE'

Cadena de identificación del servidor, dada en las cabeceras cuando se responde a peticiones.

'SERVER_PROTOCOL'

Nombre y revisión del protocolo de información mediante el cual fue solicitada la página; es decir, 'HTTP/1.0';

'REQUEST_METHOD'

Cuál método de petición fue usado para acceder a la página; es decir, 'GET', 'HEAD', 'POST', 'PUT'.

'REQUEST_TIME'

La marca de tiempo del inicio de la petición. Disponible desde PHP 5.1.0.

'QUERY_STRING'

La cadena de consulta, si existe, mediante la cual se accedió a la página.

'DOCUMENT_ROOT'

El directorio raíz de documentos bajo el que está siendo ejecutado el script actual, tal y como se define en el archivo de configuración del servidor.

'HTTP_ACCEPT'

Contenidos de la cabecera Accept: de la petición actual, si existe.

'HTTP_ACCEPT_CHARSET'

Contenidos de la cabecera Accept-Charset: de la petición actual, si existe. Ejemplo: 'iso-8859-1,*,utf-8'.

'HTTP_ACCEPT_ENCODING'

Contenidos de la cabecera Accept-Encoding: de la petición actual, si existe. Ejemplo: 'gzip'.

'HTTP_ACCEPT_LANGUAGE'

Contenidos de la cabecera Accept-Language: de la petición actual, si existe. Ejemplo: 'en'.

'HTTP_CONNECTION'
Contenidos de la cabecera Connection: de la petición actual, si existe. Ejemplo: 'Keep-Alive'.
'HTTP_HOST'

Contenidos de la cabecera Host: de la petición actual, si existe.

'HTTP_REFERER'

La dirección de la página (si la hay) la cual refirió al agente de usuario a la página actual. Este valor es definido por el agente de usuario.

'HTTP_USER_AGENT'

Contenidos de la cabecera User-Agent: de la petición actual, si existe. Esta es una cadena que denota el agente de usuario que está accediendo a la página.

'HTTPS'

Definido con un valor no-vacío si el script fue solicitado a través del protocolo HTTPS.

'REMOTE_ADDR'

La dirección IP desde donde el usuario está observado la página actual.

'REMOTE_HOST'

El nombre Host desde donde el usuario está viendo la página actual. La consulta dns de vuelta está basada en el valor REMOTE_ADDR del usuario.

'REMOTE_PORT'

El puerto que está siendo usado en la máquina del usuario para comunicarse con el servidor web.

'SCRIPT_FILENAME'

La ruta absoluta del nombre del script siendo ejecutado actualmente.

'SERVER_ADMIN'

El valor dado a la directiva SERVER_ADMIN (para Apache) en el archivo de configuración del servidor web. Si el script está siendo ejecutado en un host virtual, éste será el valor definido para ese host virtual.

'SERVER_PORT'

El puerto en el equipo servidor que está siendo usado por el servidor web para comunicación. En configuraciones predeterminadas, ese valor será '80'; usando SSL, por ejemplo, este valor cambiará a cualquiera que sea el puerto que esté definido para HTTP seguro.

'SERVER_SIGNATURE'

Cadena que contiene la versión del servidor y el nombre de host virtual que es agregado a las páginas generadas por el servidor, si está habilitada esta funcionalidad.

'PATH_TRANSLATED'

Ruta sobre el sistema de archivos (no la raíz de documentos) al script actual, luego de que el servidor haya realizado cualquier asignación al vuelo virtual-a-real.

'SCRIPT_NAME'

Contiene la ruta del script actual. Ésta es útil para páginas que necesitan apuntar a ellas mismas. La constante __FILE__ contiene la ruta completa y nombre del archivo actual (es decir, incluido).

'REQUEST_URI'

El URI que fue dado para acceder a esta página; por ejemplo, '/index.html'.

'PHP_AUTH_DIGEST'

Cuando se está corriendo bajo Apache como módulo, realizando autenticación HTTP Digest, esta variable recibe el valor de la cabecera 'Authorization' enviada por el cliente (la cual debería ser usada para efectuar la validación apropiada).

'PHP_AUTH_USER'

Cuando se corre sobre Apache o IIS (ISAPI en PHP 5) como módulo realizando autenticación HTTP, esta variable es definida con el nombre de usuario definido por el cliente.

'PHP_AUTH_PW'

Cuando se corre sobre Apache o IIS (ISAPI en PHP 5) como módulo realizando autenticación HTTP, esta variable es definida con la contraseña entregada por el usuario.

'AUTH_TYPE'

Cuando se corre sobre Apache como módulo realizando autenticación HTTP, esta variable es definida con el tipo de autenticación.

Nota: Aparecieron en 4.1.0. En versiones anteriores se debe de utilizar: *$HTTP_SERVER_VARS*

5.2.2.3 Variables de servidor: $_ENV

Estas variables son importadas en el espacio de nombres global de PHP desde el entorno bajo el que está siendo ejecutado el intérprete PHP. Muchas son entregadas por el intérprete de comandos bajo el que PHP está corriendo y diferentes sistemas suelen tener diferentes tipos de intérpretes de comandos, una lista definitiva es imposible.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo del script.

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_ENV_VARS*.

5.2.2.4 Cokkies HTTP: $_COKKIE

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script.

No necesita hacer global $_COOKIE; para acceder a ella dentro de funciones o métodos, como lo hace con *$HTTP_COOKIE_VARS*.

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_COOKIE_VARS*.

5.2.2.5 Variables HTTP GET: $_GET

Una matriz asociativa de variables pasadas al script actual a través del método HTTP GET. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática.

5.2.2.6 Variables HTTP POST: $_POST

Una matriz asociativa de variables pasadas al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script.

No necesita hacer global $_POST; para acceder a ella dentro de funciones o métodos, como lo hace con $HTTP_POST_VARS.

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_POST_VARS*.

5.2.2.7 Variables de carga de Archivos: $_FILES

Una matriz asociativa de elementos cargados al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script.

No necesita hacer global $_FILES; para acceder a ella dentro de funciones o métodos, como lo hace con $HTTP_POST_FILES.

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_POST_FILES*.

5.2.2.8 Variables de Petición: $_REQUEST

Una matriz asociativa que consiste en los contenidos de $_GET, $_POST, y $_COOKIE.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script.

No necesita hacer global $_REQUEST; para acceder a ella dentro de funciones o métodos.
Nota: Introducidas en 4.1.0. No existe una matriz equivalente en versiones anteriores.

Nota: Antes de PHP 4.3.0, la información de $_FILES también era incluida en $_REQUEST.

5.2.2.9 Variables de sesión: $_SESSION

Una matriz asociativa que contiene las variables de sesión disponibles en el script actual. Consulte la documentación sobre Funciones de Sesión para más información sobre cómo es usada ésta matriz.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script.

No necesita hacer global $_SESSION; para acceder a ella dentro de funciones o métodos, como lo hace con $HTTP_SESSION_VARS.

Nota: Introducidas en 4.1.0. En versiones anteriores, use *$HTTP_SESSION_VARS*.

5.2.2.10 Variables globales: $GLOBALS

Una matriz asociativa que contiene referencias a todas las variables que están definidas actualmente en el contexto global del script. Los nombres de las variables son las claves de la matriz.

Esta es una variable 'superglobal', o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de un script.

No necesita hacer global $_GLOBALS; para acceder a ella dentro de funciones o métodos.

Nota: *$GLOBALS* ha estado disponible desde PHP 3.0.0.

5.2.2.11 El mensaje de error previo: $php_errormsg

$php_errormsg es una variable que contiene el texto del último mensaje de error generado por PHP. Esta variable solo estará disponibles dentro del contexto en el que el error ocurrió, y solo si la opción de configuración track_errors está habilitada (por defecto está definida como off).

5.2.3 Tipos de datos

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.

5.2.3.1 Tipos de datos primitivos

- **Bolean**.- Este es el tipo más simple. Un boolean expresa un valor de verdad. Puede ser TRUE or FALSE.

- **Integer**.- Un integer es un número del conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

- **Float** (número de punto-flotante, también conocido como 'double').- El tamaño de un flotante depende de la plataforma, aunque un valor común

consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

- **String**.- Un valor string es una serie de caracteres. En PHP, un caracter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes. Esto implica también que PHP no tiene soporte nativo de Unicode. Vea utf8_encode() y utf8_decode() para conocer sobre el soporte Unicode.

5.2.3.2 Tipos de datos compuestos

- **Array**.- Una matriz en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves. Este tipo es optimizado en varias formas, de modo que puede usarlo como una matriz real, o una lista (vector), tabla asociativa (caso particular de implementación de un mapa), diccionario, colección, pila, cola y probablemente más. Ya que puede tener otra matriz PHP como valor, es realmente fácil simular árboles.

5.2.3.3 Tipos de datos especiales

- **Resuorce**.- Un recurso es una variable especial, que contiene una referencia a un recurso externo. Los recursos son creados y usados por funciones especiales. Vea el apéndice para un listado de todas estas funciones y los tipos de recurso correspondientes.

- **Null**.- El valor especial NULL representa que una variable no tiene valor. NULL es el único valor posible del tipo NULL.

5.2.4 Expresiones

Las expresiones son la piedra angular de PHP. En PHP, casi cualquier cosa que se escribe es una expresión. La forma más simple y ajustada de definir una expresión es "cualquier cosa que tiene un valor".

Las formas más básicas de expresiones son las constantes y las variables. Cuando escribes "$a = 5", estás asignando '5' a $a. '5', obviamente, tiene el valor 5 ó, en otras palabras '5' es una expresión con el valor 5 (en este caso, '5' es una constante entera).

5.2.5 Operadores

Se refieren a la teoría de la sintaxis en PHP.

5.2.5.1 Operadores Aritméticos

Los operadores de la aritmética tradicional funcionan de la misma forma con PHP.

Ejemplo:

-$a
Negación. El opuesto de $a.

$a + $b
Adición. Suma de $a y $b.

$a - $b
Substracción. Diferencia entre $a y $b.

$a * $b
Multiplicación. Producto de $a y $b.

$a / $b
División. Cociente de $a y $b.

$a % $b
Módulo. Resto de $a dividido por $b.

5.2.5.2 Operadores de Asignación

El operador básico de asignación es "=". Lo que quiere decir es que el operando de la izquierda recibe el valor de la expresión a la derecha (es decir, "se define a"). El valor de una expresión de asignación es el valor que se asigna.

En conjunto con el operador básico de asignación, existen "operadores combinados" para todos los operadores de aritmética binaria, unión de matrices y de cadenas, que permiten usar un valor en una expresión y luego definir su valor como el resultado de esa expresión.

Por ejemplo:

```
<?php

$a = 3;
$a += 5; // define $a como 8
$b = "&iexcl;Hola ";
$b .= "a todos!"; // define $b como "&iexcl;Hola a todos!", tal como $b = $b . "a todos!";

?>
```

5.2.5.3 Operadores bit a bit

Los operadores bit a bit le permiten activar o desactivar bits individuales de un entero. Si los parámetros tanto a la izquierda y a la derecha son cadenas, el operador bit a bit trabajará sobre los valores ASCII de los caracteres.

Por ejemplo:

```
<?php
echo 12 ^ 9; // Imprime '5'

echo "12" ^ "9"; // Imprime el caracter Backspace (ascii 8)
                    // ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8

echo "hallo" ^ "hello"; // Imprime los valores ascii #0 #4 #0 #0 #0
                  // 'a' ^ 'e' = #4
?>
```

5.2.5.4 Operadores de comparación

Los operadores de comparación, como su nombre indica, permiten comparar dos valores.

$a == $b
Igual. TRUE si $a es igual a $b.

$a === $b
Idéntico. TRUE si $a es igual a $b, y son del mismo tipo. (A partir de PHP 4)

$a != $b
Diferente. TRUE si $a no es igual a $b.

$a <> $b
Diferente. TRUE si $a no es igual a $b.

$a !== $b
No idénticos. TRUE si $a no es igual a $b, o si no son del mismo tipo. (A partir de PHP 4)

$a < $b
Menor que. TRUE si $a es estrictamente menor que $b.

$a > $b
Mayor que. TRUE si $a es estrictamente mayor que $b.

$a <= $b

Menor o igual que. TRUE si $a es menor o igual que $b.

$a >= $b
Mayor o igual que. TRUE si $a es mayor o igual que $b.

5.2.5.5 Operadores de control de errores

PHP ofrece soporte para un operador de control de errores: el signo de arroba (@). Cuando es colocado al comienzo de una expresión, cualquier mensaje de error que pudiera generarse a causa de esa expresión será ignorado.

Por ejemplo:

```
<?php

$valor = @$caché[$llave];
// no producirá una anotación si el índice $llave no existe.

?>
```

5.2.5.6 Operadores de ejecución

PHP soporta un operador de ejecución: las comillas invertidas (``). (No se trata de comillas sencillas). PHP intentará ejecutar el contenido entre las comillas como si se tratara de un comando del intérprete de comandos; su salida será devuelta (es decir, no será simplemente volcada como salida; puede ser asignada a una variable). El uso del operador de comillas invertidas es idéntico al de shell_exec().

Por Ejemplo:

```
<?php

$salida = `ls -al`;
echo "<pre>$salida</pre>";

?>
```

5.2.5.7 Operadores de ejecución

PHP ofrece soporte de operadores de pre- y post-incremento y decremento, estilo-C.

++$a
Pre-incremento. Incrementa $a en uno, y luego devuelve $a.

$a++
Post-incremento. Devuelve $a, y luego incrementa $a en uno.

--$a
Pre-decremento. Decrementa $a en uno, luego devuelve $a.

5.2.5.8 Operadores de Lógica

$a and $b
Y. TRUE si tanto $a como $b son TRUE.

$a or $b
O. TRUE si cualquiera de $a o $b es TRUE.

$a xor $b
O exclusivo (Xor). TRUE si $a o $b es TRUE, pero no ambos.

! $a
No. TRUE si $a no es TRUE.
$a && $b
Y. TRUE si tanto $a como $b son TRUE.

$a || $b
O.
TRUE si cualquiera de $a o $b es TRUE.

5.2.5.9 Operadores de Cadena

Existen dos operadores para datos tipo string o cadena. El primero es el operador de concatenación ('.'), el cual devuelve el resultado de concatenar sus argumentos a lado derecho e izquierdo. El segundo es el operador de asignación sobre concatenación ('.='), el cual adiciona el argumento del lado derecho al argumento en el lado izquierdo.

Por Ejemplo:

```
<?php

$a = "&iexcl;Hola ";
$b = $a . "Mundo!"; // ahora $b contiene "&iexcl;Hola Mundo!"

$a = "&iexcl;Hola ";
$a .= "Mundo!";    // ahora $a contiene "&iexcl;Hola Mundo!"

?>
```

5.2.5.10 Operadores de Matrices

$a + $b
Unión. Unión de $a y $b.

$a == $b
Igualdad. TRUE si $a y $b tienen las mismas parejas llave/valor.

$a === $b
Identidad. TRUE si $a y $b tienen las mismas parejas llave/valor en el mismo orden y de los mismos tipos.

$a != $b
No-igualdad. TRUE si $a no es igual a $b.

$a <> $b
No-igualdad. TRUE si $a no es igual a $b.

$a !== $b
No-identidad. TRUE si $a no es idéntico a $b.

5.2.5.11 Operadores de Tipo

PHP tiene un operador único de tipo: instanceof es usado para determinar si un objeto dado, sus padres o sus implementaciones de interfaces son de una clase de objeto especificada. El operador instanceof fue introducido en PHP 5. Antes de esta versión, is_a() era utilizado, pero is_a() ha sido marcado como obsoleto desde entonces en favor de instanceof.

Por Ejemplo:

```
<?php

class A { }
class B { }

$cosa = new A;

if ($cosa instanceof A) {
   echo 'A';
}
if ($cosa instanceof B) {
   echo 'B';
}

?>
```

Dado que $cosa es un objeto de tipo A, pero no de tipo B, sólo el bloque dependiente del tipo A será ejecutado: A

5.2.6 Bloques y Sentencias

Todo script PHP se compone de una serie de sentencias. Una sentencia puede ser una asignación, una llamada a función, un bucle, una sentencia condicional e incluso una sentencia que no haga nada (una sentencia vacía). Las sentencias normalmente acaban con punto y coma.

Las sentencias se pueden agrupar en grupos de sentencias encapsulándolas con llaves. Un grupo de sentencias es también una sentencia.

5.2.6.1 IF

La construcción if es una de las más importantes características de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código.

```
<?php

if (expr)
   sentencia

?>
```

5.2.6.2 ELSE

else extiende una sentencia if para ejecutar una sentencia en caso de que la expresión en la sentencia if se evalúe como FALSE.

Por Ejemplo:

```
<?php

 if ($a > $b) {
    print "a es mayor que b";
 } else {
    print "a NO es mayor que b";
 }

?>
```

5.2.6.3 ELSEIF

elseif, como su nombre sugiere, es una combinación de if y else. Como else, extiende una sentencia if para ejecutar una sentencia diferente en caso de que la expresión if original se evalúa como FALSE. No obstante, a diferencia de else, ejecutará esa expresión alternativa solamente si la expresión condicional elseif se evalúa como TRUE.

5.2.6.4 WHILE

Los bucles while son los tipos de bucle más simples en PHP. Se comportan como su contrapartida en C. La forma básica de una sentencia while es:

while (expr) sentencia

5.2.6.5 DO - WHILE

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

5.2.6.6 FOR

Los bucles for son los bucles más complejos en PHP. Se comportan como su contrapartida en C. La sintaxis de un bucle for es:

for (expr1; expr2; expr3) sentencia

5.2.6.7 FOREACH

PHP 4 (PHP3 no) incluye una construcción foreach, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre matrices. foreach funciona solamente con matrices y devolverá un error si se intenta utilizar con otro tipo de datos ó variables no inicializadas. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

foreach(expresion_array as $value) sentencia
foreach(expresion_array as $key => $value) sentencia

La primera forma recorre el array dado por expresion_array. En cada iteración, el valor del elemento actual se asigna a $value y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable $key en cada iteración.

5.2.6.8 BREAK

break escapa de la estructuras de control iterante (bucle) actuales for, while, o switch.

break acepta un parámetro opcional, el cual determina cuantas estructuras de control hay que escapar.

5.2.6.9 CONTINUE

continue se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

5.2.6.10 SWITCH

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia switch.

5.2.6.11 DECLARE

La construcción declare es usada para definir directivas de ejecución para un bloque de código. La sintaxis de declare es similar a la de las otras estructuras de control:

declare (directiva) sentencia

5.2.6.12 RETURN

Si se llama desde una función, return() termina inmediatamente la ejecución de la función y retorna su argumento como valor de la función. return() también terminará la ejecución de una sentencia eval()ó un script PHP.

5.2.6.13 REQUIRE

require() incluye y evalúa el archivo especificado.

5.2.6.14 INCLUDE

La sentencia include() incluye y evalúa el archivo especificado.

Esta documentación también se aplica a la función require(). require() y include() son idénticas en todos los aspectos excepto en el modo de actuar ante un error.

5.2.6.15 REQUIRE_ONCE

La función require_once() incluye y evalúa el fichero especificado durante la ejecución del script. Se comporta de manera similar a require(), con la única diferencia que si el código ha sido ya incluido, no se volverá a incluir.

5.2.6.16 INCLUDE_ONCE

La función include_once() incluye y evalúa el fichero especificado durante la ejecución del script. Se comporta de manera similar a include(), con la única diferencia que si el código ha sido ya incluido, no se volverá a incluir[2].

[2]Copyright © 2003-2004 por el Grupo de documentación de PHP

6. INTERACCIÓN DE WWW CON BASES DE DATOS

6.1 Objetivo

Se conocerá y desarrollará una aplicación de bases de datos que funcione a través del WWW empleando herramientas de software libre.

6.2 Introducción

La relación entre PHP y bases de datos, especialmente MySQL y Postgres, es muy estrecha y beneficiosa. De hecho, cuando se habla de Web y PHP, es muy difícil que no se mencione también a una base de datos. Después de todo, el Web está pensado para almacenar y permitir los accesos a cantidades enormes de información. Mientras mayor sea la cantidad de información y más alta la frecuencia de actualización de un sitio Web, mayor es su valor y sus ventajas sobre otros medios.

Tal vez la mayor ventaja de PHP sobre sus competidores es la integración con los sistemas de bases de datos y el soporte nativo a las distintas bases de datos existentes, libres y comerciales.

Las razones principales para usar una base de datos son:

- Evitar redundancias.
- Evitar programas complicados.
- Búsquedas.
- Seguridad.
- Arquitectura n-tier

SQL

SQL, "Structured Query Language" representa un método estricto y más general de almacenamiento de datos que estándares anteriores. SQL es un estándar ANSI (www.ansi.org) y ECMA (www.ecma.ch).

La estructura básica de una base de datos relacional con SQL es muy simple.

Una instalación de base de datos puede contener múltiples bases de datos, cada base de datos puede contener un conjunto de tablas. Cada tabla está compuesta de un conjunto de columnas cuidadosamente diseñadas y cada elemento (o entrada) de la tabla es una fila.

6.2.1 Características de MySQL

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas Web con contenido dinámico, justamente por su simplicidad; aquellos elementos faltantes fueron llenados por la vía de las aplicaciones que la utilizan.

Poco a poco los elementos faltantes en MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto[1].

6.3 Manejo de formularios como Front-End

En muchos casos, los términos front end y back end se refieren al principio y final de un proceso. Estos términos adquieren una relevancia mayor en ciertas áreas particulares.

En diseño de software, el front-end es la parte del software que interactúa con el usuario y el back-end es la parte que procesa la entrada desde el front-end. La separación del sistema en "front ends" y "back ends" es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas.

La idea general es que el front-end es el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas y procesarlas de una manera conforme a la especificación que el back-end pueda usar. La conexión del front-end y el back-end es un tipo de interfaz.
En diseño web (o desarrollo web), hace referencia a la visualización del usuario navegante (por un lado), y del administrador del sitio con sus respectivos sistemas (por el otro).

Muchos métodos conocidos de interactuar con computadoras pueden ser conceptualizados en términos de "front-end" y "back-end". Por ejemplo, un administrador de archivos gráfico como puede ser Windows Explorer o el Nautilus

[1] Página oficial de MySQL

file manager pueden ser considerados como un front-end para el file system de la computadora.

6.3.1 Introducción a las etiquetas HTML de los formularios

Los formularios son definidos por medio de las etiquetas <form> y </form>.

Entre estas dos etiquetas colocaremos todos los campos y botones que componen el formulario. Dentro de la etiqueta <form> debemos especificar algunos atributos:

action: Define el tipo de acción a llevar a cabo con el formulario. Como ya hemos dicho, existen dos posibilidades:

- El formulario es enviado a una dirección de correo electrónico
- El formulario es enviado a un programa o script que procesa su contenido

method: Este atributo se encarga de especificar la forma en la que el formulario es enviado. Los dos valores posibles que puede tomar este atributo son post y get. Para efectos prácticos.

enctype: Se utiliza para indicar la forma en la que viajará la información que se mande por el formulario. En el caso más corriente, enviar el formulario por correo electrónico, el valor de este atributo debe de ser "text/plain". Así conseguimos que se envíe el contenido del formulario como texto plano dentro del email.

Si queremos que el formulario se procese automáticamente por un programa, generalmente no utilizaremos este atributo, de modo que tome su valor por defecto, es decir, no incluiremos enctype dentro de la etiqueta <form>.

6.3.2 Manejo inicial de los campos de la base de datos desde formularios

Texto

Las cajas de texto son colocadas por medio de la etiqueta <input>. Dentro de esta etiqueta hemos de especificar el valor de dos atributos: type y name.

La etiqueta es de la siguiente forma:

<input type="text" name="nombre">

De este modo expresamos nuestro deseo de crear una caja de texto cuyo contenido será llamado nombre.

El nombre del elemento del formulario es de gran importancia para poder identificarlo en nuestro programa de procesamiento o en el mail recibido. Por otra

parte, es importante indicar el atributo type, ya que, existen otras modalidades de formulario que usan esta misma etiqueta.

El empleo de estas cajas esta fundamentalmente destinado a la toma de datos breves: palabras o conjuntos de palabras de longitud relativamente corta.

Además de estos dos atributos, esenciales para el correcto funcionamiento de la etiqueta, existen otra serie de atributos que pueden resultarnos de utilidad pero que no son imprescindibles:

size: Define el tamaño de la caja en número de caracteres. Si al escribir el usuario llega al final de la caja, el texto ira desfilando a medida que se escribe haciendo desaparecer la parte de texto que queda a la izquierda.

maxlength: Indica el tamaño máximo del texto que puede ser tomado por el formulario. Es importante no confundirlo con el atributo size. Mientras el primero define el tamaño aparente de la caja de texto, maxlength indica el tamaño máximo real del texto que se puede escribir. Podemos tener una caja de texto con un tamaño aparente (size) que es menor que el tamaño máximo (maxlength). Lo que ocurrirá en este caso es que, al escribir, el texto ira desfilando dentro de la caja hasta que lleguemos a su tamaño máximo definido por maxlength, momento en el cual nos será imposible continuar escribiendo.

value: En algunos casos puede resultarnos interesante asignar un valor definido al campo en cuestión. Esto puede ayudar al usuario a rellenar más rápidamente el formulario o darle alguna idea sobre la naturaleza de datos que se requieren. Este valor inicial del campo puede ser expresado mediante el atributo value.

Texto oculto

Se puede esconder el texto escrito por medio de asteriscos para aportar una cierta confidencialidad. Este tipo de campos son análogos a los de texto con una sola diferencia: remplazamos el atributo type="text" por type="password":

<input type="password" name="nombre">

Estos campos son ideales para la introducción de datos confidenciales, principalmente códigos de acceso.

Texto largo

Si deseamos poner a la disposición de usuario un campo de texto donde pueda escribir cómodamente sobre un espacio compuesto de varias líneas, se puede invocar a la etiqueta: <textarea>.

Este tipo de campos son prácticos cuando el contenido a enviar no es un nombre teléfono o cualquier otro dato breve, sino más bien, un comentario, opinión, etc.

Dentro de la etiqueta textarea deberemos indicar el atributo *name* para asociar el contenido a un nombre que será asemejado a una variable en los programas de proceso. Además, se pueden definir las dimensiones del campo a partir de los atributos siguientes:

rows: Define el número de líneas del campo de texto.

cols: Define el número de columnas del campo de texto.

La etiqueta queda por tanto de esta forma:

```
<textarea name="comentario" rows="10" cols="40"></textarea>
```

Listas de opciones

Las listas de opciones son ese tipo de menús desplegables que nos permiten elegir una o varias de las múltiples opciones que nos proponen. Para construirlas se emplea la etiqueta: <select>.

Se podrá definir su nombre por medio del atributo name. Cada opción será incluida en una línea precedida de la etiqueta <option>.

Se puede ver, a partir de estas directivas, la forma más típica y sencilla de esta etiqueta:

```
<select name="estacion">
        <option>Primavera</option>
        <option>Verano</option>
        <option>Otoño</option>
        <option>Invierno</option>
</select>
```

size: Indica el número de valores mostrados de la lista. El resto pueden ser vistos por medio de la barra lateral de desplazamiento.

multiple: Permite la selección de más varios elementos de la lista. Este atributo se expresa sin valor alguno, es decir, no se utiliza con el igual: simplemente se pone para conseguir el efecto, o no se pone si queremos una lista desplegable común.

Botones de radio.

Existe otra alternativa para plantear una elección, en este caso, obligamos al usuario a elegir únicamente una de las opciones que se le proponen.

La etiqueta empleada en este caso es <input> en la cual tendremos el atributo type ha de tomar el valor radio:

```
<input type="radio" name="estacion" value="1">Primavera
<br>
<input type="radio" name="estacion" value="2">Verano
<br>
<input type="radio" name="estacion" value="3">Otoño
<br>
<input type="radio" name="estacion" value="4">Invierno
```

Cajas de validación

Este tipo de elementos pueden ser activados o desactivados por el visitante por un simple clic sobre la caja en cuestión. La sintaxis utilizada es muy similar a las vistas anteriormente:

```
<input type="checkbox" name="casado" value=”1”>Sí
<input type="checkbox" name="casado" value=”0”>No
```

La única diferencia fundamental es el valor adoptado por el atributo type.

6.4 Instalación y configuración de la base de datos en Linux

Para ejemplificar instalaremos Postgresql en Linux como se llevo a cabo durante el Diplomado.

INSTALACIÓN DE POSTGRESQL

1. Configuración.

El primer paso del proceso de instalación es la configuración del entorno de postgresql y para ello deberá trabajar la siguiente sintaxis desde el shell:

```
./configure
```

2. Construcción.

Para iniciar la construcción deberá teclear lo siguiente:

```
gmake
```

3. Asegurar el administrador.

Para asegurar de que se trata de root deberás escribir lo siguiente:

su

4. Prueba de Regresión.

Si deseas probar la nueva construcción de instalación del servidor antes de instalarlo, podrás ejecutar la prueba de regresión de la siguiente manera:

gmake check

5. Instalando los archivos.

Nota: Si estas actualizando un sistema existente deberás instalar los nuevos archivos sobre los anteriores y deberás tener un back up de las bases de datos existentes.

Para instalar postgresql teclea lo siguiente:

gmake install

6. Crear una cuenta para postgresql.

Crea una cuenta para el servidor de postgresql. Este será el usuario que levantará el servidor. Para el uso en producción deberías crear una cuenta sin privilegios (comúnmente se usa postgres).

adduser postgres

6. Crear la base de datos de instalación.

Crea la instalación de la base de datos de sistema con el commando initdb, para lo cual antes deberás cambiarte a la cuenta de postgres, ya que no funcionará como root. Además deberá indicar la construcción del directorio de datos con los privilegios para postgres.

mkdir /usr/local/pgsql/data

chown postgres /usr/local/pgsql/data

su – postgres

/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data

7. Arranque del servidor.

El paso anterior debería también indicarle al final como iniciar su servidor de bases de datos. El comando que hace lo mismo se describe a continuación:
/usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &

8. Crear la base de datos

Deberá crear una base de datos para poder interactuar con postgresql y posteriormente podrá utilizar el cliente de sql. Para ello deberá seguir las siguientes instrucciones.

/usr/local/pgsql/bin/createdb test

/usr/local/pgsql/bin/psql test

Notas:

Para arrancar el servidor y se desea además asegurar el puerto de servicio se deberá trabajar la siguiente sintaxis:

/usr/local/pgsql/bin/postmaster -i -p 5432-D /usr/local/pgsql/data &

Otra forma de arrancarlo es utilizar el pg_ctl de la siguiente forma:

/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data start

Y para detener el servicio del servidor la sintaxis es la siguiente:

/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop

6.4.1 Comparación entre las bases de datos gratuitas para Linux

Comparación entre Mysql y Postgresql

MySQL:

Su principal objetivo de diseño fue la VELOCIDAD. Se sacrificaron algunas características esenciales en sistemas más "serios" con este fin.

Otra característica importante es que consume MUY POCOS RECURSOS, tanto de CPU como de memoria.

Licencia GPL a partir de la versión 3.23.19.

Ventajas:

- Mayor rendimiento. Mayor velocidad tanto al conectar con el servidor como al servir selects y demás.

- Mejores utilidades de administración (backup, recuperación de errores, etc.).

- Aunque se cuelgue, no suele perder información ni corromper los datos.
- Mejor integración con PHP.

- No hay límites en el tamaño de los registros.

- Mejor control de acceso, en el sentido de que usuarios tienen acceso a que tablas y con que permisos.

- MySQL se comporta mejor que Postgres a la hora de modificar o añadir campos a una tabla "en caliente".

Inconvenientes:

- No soporta transacciones, "roll-backs" ni subselects.

- No considera las claves ajenas. Ignora la integridad referencial, dejándola en manos del programador de la aplicación.

PostgreSQL:

Postgres intenta ser un sistema de bases de datos de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase.

Licencia BSD.

Ventajas:

- Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.

- Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial).

- Tiene mejor soporte para triggers y procedimientos en el servidor.

- Soporta un subconjunto de SQL92 MAYOR que el que soporta MySQL. Además tiene ciertas características orientadas a objetos.

Inconvenientes:

- Consume BASTANTES más recursos y carga mas el sistema.
- Limite del tamaño de cada fila de las tablas a 8k!!! (se puede ampliar a 32k recompilando, pero con un coste añadido en el rendimiento).
- Es de 2 a 3 veces más lenta que MySQL.
- Menos funciones en PHP.

En cuanto a consideraciones de estabilidad del servidor, cada comparativa de datos contradictorios. En general parece que MySQL es más estable (aunque también hay gente que opina lo contrario), y que Postgres tiende a desperdiciar memoria y sobrecargar bastante el sistema (aunque de nuevo, hay opiniones distintas).

Como conclusión a la comparación entre MySQL y Postgres, parece aceptado que MySQL junto con Apache y PHP forman un buen equipo para servir páginas Web con contenido dinámico, discusiones, noticias, etc., por ejemplo al estilo de SlashDot. En general, sistemas en los que la velocidad y el numero de accesos concurrentes sea algo primordial, y la seguridad no sea muy importante (pueda bastar con hacer backups periódicos que se restauraran tras una Saida del servidor). En cambio, para sistemas más serios en las que la consistencia de la BD sea fundamental (BD con información realmente importante, bancos, etc.) PostgreSQL es una mejor opción pese a su mayor lentitud.

6.4.1.1 PostgrSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley.

El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por Defense Advanced Research Projects Agency (DARPA), el Army Research Office (ARO), el opcional Science Foundation (NSF), y ESL, Inc.

PostgreSQL es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales.

PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto,

PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Las principales características de este gestor de bases de datos son:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP …), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

6.4.1.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo licencia GPL de GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras

herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

6.4.2 Instalación y configuración de la base de datos

Se debe de conseguir las fuentes de MySQL. De la página oficial:

http://www.mysql.com/

Con el tar se descomprimen las fuentes. Se crea con mkdir el directorio de instalación para MySQL.

Al descomprimir las fuentes se crea el directorio mysql-5.0.13-rc, se procede a ingresar a este directorio con cd:

$ cd mysql-5.0.13-rc/

Opciones de config:

--prefix = La ruta del directorio donde vamos a instalar nuestro MySQL.

--sysconfdir = La ruta del directorio donde se va a alojar nuestro archivo de configuración (my.cnf).

--with-tcp-port = El puerto por donde va a escuchar nuestro servidor MySQL, por defecto es 3306 por lo que tocará colocar otro puerto.

--with-mysqld-user = El usuario que arrancará el demonio mysqld.

--with-unix-socket-path = La ruta absoluta del socket UNIX. Cuandos las conexiones al servidor son en un ambiente local MySQL utiliza un socket en vez del puerto TCP.

Después se procede a ejecutar $make; make install[2].

[2] Página oficial de MySQL

7. INTRODUCCIÓN A LA SEGURIDAD EN CÓMPUTO

7.1 Objetivo

Se reconocerá la importancia de la seguridad e identificará los elementos que le permitirán proteger el sistema y la información.

7.2 Introducción

La seguridad implica compromisos. Casi siempre el compromiso es entre la seguridad y la facilidad de uso. No hay un sistema totalmente seguro que pueda ser utilizable. El objetivo es determinar que tan valiosos son los datos y el tiempo del usuario, y que tanta protección se desea. La seguridad es un proceso progresivo, que no puede realizarse de la noche a la mañana. Al mejorar la seguridad por etapas se puede llegar al balance adecuado.

7.2.1 ¿Qué es seguridad en Cómputo?

La seguridad de un sistema de cómputo se da cuando:

- Hay confianza en él
- El comportamiento del software es el esperado
- La información almacenada

  • Inalterada
  • Accesible

7.2.2 Conceptos de seguridad

Un servicio de seguridad es una característica que debe tener un sistema para satisfacer una política de seguridad.

El estándar ISO 7498-2, que cubre las comunicaciones seguras entre sistemas abiertos define cinco clases de servicios de seguridad.

7.2.2.1 Autentificación

Se asegura de la identidad de la persona del otro lado de la línea, se puede realizar mediante alguno de los siguientes mecanismos: “algo que se sabe”, “algo que se tiene” o “algo que se es”.

7.2.2.2 Confidencialidad

Solo el propietario del secreto es capaz de descifrar la información.

7.2.2.3 Disponibilidad

Asegurar que el sistema este disponible cuando se le requiera.

7.2.2.4 Integridad

Se asegura que la información no ha cambiado durante la transmisión.

7.2.2.5 No Repudio

Se asegura que el emisor de la información no pueda negar haberla enviado.

7.3 Control de Acceso

7.3.1 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)

7.3.2 Mecanismos de Control de Acceso

Si hay un número significativo de usuarios dentro de una organización, los administradores de sistemas tienen dos opciones básicas disponibles para forzar el uso de buenas contraseñas. Ellos pueden crear contraseñas para el usuario o dejar que los usuarios crean sus propias contraseñas, a la vez que verifican que las contraseñas sean de calidad aceptable.

Al crear las contraseñas para los usuarios asegura que las contraseñas sean buenas, pero se vuelve una tarea agotadora a medida que la organización crece. También incrementa el riesgo de que los usuarios escriban sus contraseñas en papel.

Por estas razones, la mayoría de los administradores de sistemas prefieren dejar que los usuarios creen sus propias contraseñas, pero activamente verifican que las contraseñas sean buenas y, en algunos casos, obligan a los usuarios a cambiarlas periódicamente haciéndolas caducar.

- Mecanismos de control de acceso
- Usados para proteger objetos del sistema (archivos, recursos)
- Controlan todos los tipos de acceso sobre el objeto por parte de cualquier entidad del sistema

- Dos tipos discrecional (DAC) y mandatorio/obligatorio (MAC)

La autenticación pretende establecer quién eres, establece qué puedes hacer con el sistema.

Existen dos modelos: DAC y 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 mandatorio (MAC): Tiene una etiqueta de seguridad. Es el sistema quién protege los recursos.

Para propósitos de seguridad, el programa de instalación configura el sistema para usar el Message-Digest Algorithm (MD5) y contraseñas shadow. Se recomienda que no se cambien estas configuraciones.

Si se quita la selección de MD5 durante la instalación, se utilizará el formato más viejo Data Encryption Standard (DES). Este formato limita las contraseñas a ocho caracteres alfanuméricos (no permite caracteres de puntuación o especiales) y proporciona un nivel encriptación modesto de 56-bits.

Si se deseleccionan las contraseñas shadow durante la instalación, todas las contraseñas son almacenadas como hash de una sola vía en el archivo `/etc/passwd`, lo que hace al sistema vulnerable a ataques de piratas fuera de línea. Si un intruso puede obtener acceso a la máquina como un usuario regular, puede también copiar el archivo `/etc/passwd` a su propia máquina y ejecutar cualquier cantidad de programas de descifrado de contraseñas contra el. Si hay una contraseña insegura en el archivo, es sólo una cosa de tiempo antes de que el maleante informático la descubra.
Las contraseñas shadow eliminan este tipo de ataques almacenando los hash de las contraseñas en el archivo `/etc/shadow`, el cual sólo es leído por el usuario root.

Esto obliga al atacante potencial a intentar descubrir la contraseña remotamente mediante la conexión a un servicio de la red en la máquina, tal como SSH o FTP. Este tipo de ataques de fuerza bruta son mucho más lentos y dejan rastros obvios, pues los intentos fallidos de conexión son registrados a los archivos del sistema. Por supuesto, si el maleante o cracker comienza un ataque durante la noche y usted tiene contraseñas débiles, éste podría obtener acceso antes del amanecer y editar el archivo de registro para borrar sus rastros.

Más allá de los detalles sobre el formato y almacenamiento, está el problema del contenido. La cosa más importante que un usuario puede hacer para proteger su cuenta contra un ataque de piratas, es crear una contraseña robusta[1].

[1]http://www.seguridad.unam.mx/

7.3.3 ¿Cómo elegir contraseñas?

Cree contraseñas de al menos ocho caracteres — Mientras más larga sea la contraseña, mejor. Si está usando contraseñas MD5, debería ser de 15 caracteres de largo o más. Con las contraseñas DES, use el largo máximo (ocho caracteres).

Para crear una contraseña segura:

- Mezcle letras mayúsculas y minúsculas para reforzar su contraseña.

- Mezcle letras y números. Agregando números a las contraseñas, especialmente cuando se añaden en el medio (no solamente al comienzo o al final), puede mejorar la fortaleza de su contraseña.

- Incluya caracteres no alfanuméricos. Los caracteres especiales tales como &, $, y > pueden mejorar considerablemente su contraseña (esto no es posible si esta usando contraseñas DES).

- Seleccione una contraseña que pueda recordar. La mejor contraseña en el mundo será de poca utilidad si usted no puede recordarla. Por lo tanto utilice acrónimos u otros dispositivos nemónicos que lo ayuden a memorizar las contraseñas.

Con todas estas reglas, puede parecer difícil crear una contraseña que reúna todos estos requisitos para las buenas contraseñas a la vez que se evitan los rasgos de las malas. Afortunadamente, hay algunos pasos que uno puede tomar para generar una contraseña segura y fácil de recordar.

8. DESARROLLO DE APLICACIONES POSTGRESQL Y PHP

8.1 Objetivo

Se crearán aplicaciones dinámicas e interactivas de bases de datos para Internet, con técnicas avanzadas de lenguaje PHP y la base de datos PostgreSQL.

8.2 Introducción

PostgreSQL es un sistema de administración de base de datos objeto-relacional (ORDBMS, por sus siglas en inglés) basado en Postgres v4.2 desarrollado en la Universidad de California en el Departamento de Ciencias de la Computación de Berkeley.

PostgreSQL es un descendiente de código abierto de este código original de Berkeley. Soporta SQL92 y SQL99 y ofrece muchas características modernas:

Consultas complejas

Foreign keys

Triggers

Vistas

Integridad transaccional

Control de concurrencia multiversión.

A su vez, PostgreSQL puede ser extendido por el usuario en múltiples formas; por ejemplo, agregando nuevos tipos de datos, funciones, operadores, métodos de indexación, funciones de agregación y lenguajes procedurales.

Además, debido a la licencia libre, PostgreSQL puede ser usado, modificado y distribuido libre de cargos para cualquier propósito, sea privado, comercial o académico.

8.3 Programación Orientada a Objetos

Cuando hablamos de software OO los objetos casi siempre son elementos físicos, como puede ser un cliente, proveedor, etc. o elementos conceptuales que existen en el entorno software, por ejemplo un objeto encargado del mantenimiento de archivos. El objetivo es representar a estos elementos de la vida real y a los conceptuales como unidades de software.

La programación OO esta pensada para construir objetos que contienen atributos y operaciones de manera que cubran nuestras necesidades. Los atributos son variables que contienen información del estado de un objeto. Y las operaciones también conocidas como métodos, funciones y acciones realizan modificaciones del propio objeto o realizan alguna acción externa a éste.

Una de las principales ventajas de la programación OO es el concepto de encapsulación, conocido también como protección de datos, mediante el cual solo se pueden modificar los datos de un objeto accediendo a través de sus métodos u operaciones (interfaz del objeto). Nunca se pueden modificar directamente desde la aplicación principal.

La funcionalidad de un objeto esta sujeta a los datos que este maneja, una ventaja de usar objetos es que podemos modificar la funcionalidad de éste, añadir mejoras o corregir errores sin necesidad de cambiar su interfaz. Ya que en caso contrario un proyecto estaría sujeto a un mayor número de fallos y los cambios serían más costosos.

En algunas áreas de la programación de aplicaciones Web el uso de la programación OO está desestimada, usándose una metodología estructurada basada en funciones, esto es debido a que determinados proyectos no son lo suficientemente extensos como para aplicarles una metodología OO.

En la programación OO los objetos son únicos y son instancias a una clase determinada. En principio se define la clase con los atributos y métodos correspondientes y luego se crea el objeto que esta basado en una determinada clase (esto se conoce como instancia). Se puede comparar a un objeto con una variable y la clase sería un tipo de dato definido por nosotros.

8.3.1 Objetos y Clases

Una clase es una colección de variables y de funciones que acceden a ellas.

Las clases son tipos, es decir, son plantillas para variables. Se tiene que crear un objeto del tipo deseado con el operador new.

Las Clases pueden ser extensiones de otras clases. Las clases extendidas o derivadas tienen todos los atributos y métodos. La herencia múltiple no está soportada en PHP.

8.3.2 Constructores y Destructores

Los constructores de un objeto permiten inicializar los atributos del mismo al ser ejecutado.

Los constructores de la clase hijo, no ejecutan el constructor de la clase padre de forma explicita, se debe de llamar por medio de la siguiente instrucción:

parent::__construct()

```
<?php
class BaseClass {
  function __construct() {
     print "In BaseClass constructor\n";
  }
}

class SubClass extends BaseClass {
  function __construct() {
     parent::__construct();
     print "In SubClass constructor\n";
  }
}

$obj = new BaseClass();
$obj = new SubClass();
?>
```

Los destructores en PHP son similares en su concepto a los de lenguajes como C++, el método destructor será llamado cuando todas las referencias al objeto sean removidas o cuando el objeto sea explícitamente destruido.

```
<?php
class MyDestructableClass {
  function __construct() {
     print "In constructor\n";
     $this->name = "MyDestructableClass";
  }

  function __destruct() {
     print "Destroying " . $this->name . "\n";
  }
}

$obj = new MyDestructableClass();
?>.
```

8.3.3 Visibilidad

La visibilidad de un atributo o método, puede ser definida anteponiendole alguna de las palabras reservadas: *public*, *protected* o *private*.

Los elementos declarados como public, podrán ser utilizados en cualquier momento.

Los elementos declarados como protected, podrán ser llamados desde las clases heredadas y la misma clase.

Los elementos declarados como private, podrán ser utilizados sólo en la clase que lo define.

8.3.4 Sobrecarga

COMO SOBRESCRIBIR MÉTODOS Y ATRIBUTOS

Como hemos visto anteriormente, una subclase declara atributos nuevos y operaciones sobre una superclase. Es posible y en muchos casos útil sobrescribir las mismas operaciones o atributos declarados en la superclase. Esto se hace para dar a un atributo un valor diferente en la subclase que el que tiene en la superclase o en el caso de las operaciones para cambiar la funcionalidad de estas. Veamos un ejemplo, por ejemplo si tenemos la clase A:

```
class A {

   var $atributo = 'valor inicial'

   function operación() {

      echo 'Clase A:<br>';

      echo 'El valor de \$atributo es $this->atributo<br>';

   }

}
```

Queremos crear una subclase B y alterar el valor de atributo y la funcionalidad de operación de la clase A, entonces escribimos:

```
Class B extends A {

  var  $atributo = 'valor cambiado';

  function operación() {

    echo 'Clase B:<br>';

    echo 'El valor de \$atributo ahora es $this->atributo<br>';

  }

}
```

Como podemos observar hemos definido una variable y una operación en B con el mismo nombre que tenían en A.

Como se ha comentado anteriormente aunque declaremos B no afecta a la definición de A, y si creamos un objeto de la superclase A esté mantendrá sus valores originales. Solo sobrescribiremos los valores y funcionalidad de A cuando creemos un objeto de la clase B.

A diferencia de otros lenguajes OO, PHP no nos permitirá sobrescribir una función o atributo y poder llamar a los valores de la clase padre.

La herencia puede tener muchas capas de profundidad, por ejemplo podemos tener una clase C que es subclase de B y está última ser subclase de A, la subclase C heredará y sobrescribirá aquellos atributos y métodos de sus clases padres, en este caso A y B.

8.4 Funciones de PHP para PostgreSQL

La base de datos PostgreSQL es un producto Open Source y disponible sin costo. Postgres, desarrollado originalmente en el Departamento de Ciencias de Computación de UC Berkeley, fue pionero en muchos de los conceptos de objetos y relacionales que ahora están apareciendo en algunas bases de datos comerciales. Provee soporte para lenguajes SQL92/SQL99, transacciones, integridad referencial, procedimientos almacenados y extensibilidad de tipos. PostgreSQL es un descendiente de código abierto de su código original de Berkeley.

8.4.1 pg_affected_rows

Regresa el número de registros afectados por una instrucción SQL INSERT, UPDATE o DELETE. Sino hubiera habido registros afectados, regresará 0.

8.4.2 pg_Connect

Abre una conexión a PostgreSQL y devuelve un índice a dicha conexión en caso de tener éxito, en caso de no poder realizar la conexión regresa FALSE.

Cada uno de los argumentos deberá ser una cadena entrecomillada, incluyendo el número del puerto.

Se podrán tener múltiples conexiones abiertas.

Ej.

$conn = pg_connect("dbname=marliese port=5432");

8.4.3 pg_Close

Cierra una connexion a postgreSQL asociada al índice pasado como parámetro.

Devolverá FALSE si no se le da un índice de conexión válido y TRUE en cualquier otro caso.

8.4.4 pg_execute

Envía la petición para ejecutar una sentencia SQL y esperará por el resultado.

8.4.5 pg_Fetch_Array

Obtiene un registro en forma de un arreglo o FALSE sino hubo registros.

Además de almacenar los datos en los índices numéricos del arreglo resultante, también almacena los datos de forma de un arreglo asociativo, empleando para esto el nombre del campo como llave asociativa del arreglo.

8.4.6 pg_Fetch_Object

Obtiene un registro en forma de Objeto o FALSE sino hubo registros.

Se podrá acceder a los registros por medio de los nombres de los campos como atributos de objeto.

8.4.7 pg_fetch_result

Regresa los valores de un resultado generado por pg_query ().

8.4.8 pg_free_result

Libera la memoria y los datos asociados con un específico resultado.

Esta función se utiliza sólo si se consume demasiada memoria durante la ejecución de una consulta SQL, de otra forma la memoria es liberada automáticamente.

8.4.9 pg_num_fields

Regresa el número de campos obtenidos en una consulta.
En caso de no obtener la consulta registros la función regresará -1.

8.4.10 pg_num_rows

Regresa el número de registros obtenidos en una consulta.
En caso de no obtener registros, regresará -1.

8.4.11 pg_query

Ejecuta una sentencia SQL.
Si ocurre un error regresará FALSE. Si no se especifica un índice de conexión, se utilizará la última conexión abierta[1].

[1] *An Introduction to Database Systems* , Date, 1994 , 6, C. J. Date, 1, 1994, Addison-Wesley, 1994.

CONCLUSIONES

El uso de tecnologías asociadas al desarrollo de software libre es cada vez de mayor relevancia, gobiernos de muchos lugares han implementado sus sistemas dentro de este tipo de aplicaciones, debido al bajo costo que ello implica, sin sacrificar, claro está, el performance de lo que se requiere implementar. Que sea gratuito o abierto o libre o como sea que se le quiera llamar, que además habría que dejar claro que no existe solamente un tipo de licencia para el llamado Software Libre, no implica que sea malo o que se sacrifique la calidad de código, al contrario, tan no es así, que muchas de las empresas que cuentan con un gran peso en el campo de la informática como IBM, DELL, Novell, incluso Google, han hecho desarrollos que resultan en un nuevo proyecto de software libre o que están implementados bajo sistemas operativos Linux o que retoman aplicaciones de código abierto para la base de un nuevo proyecto.

Se puede decir acerca de los proyectos de software libre que, en donde toman su verdadera riqueza, es que muchas personas pueden ser participes del desarrollo y sobre todo, del mantenimiento de dichos proyectos, cuanto más personas se involucran en ellos, más ricos se vuelven así como el potencial de desarrollo y soporte se incrementan de forma notable.

Las aplicaciones para Web creadas con PHP como lenguaje de programación y algún manejador de base de datos como son Mysql o PostgreSQL han llegado a ser de una popularidad trascendente en el entorno de desarrollos Web, no sólo por su fácil implementación y rápida curva de aprendizaje sino por que son herramientas fiables y seguras.

Los conocimientos adquiridos mediante los módulos que el diplomado comprende son suficientes para comenzar el desarrollo e implementación de sistemas bajo el esquema de Software libre y Linux y aprovechar las ventajas que tienen las herramientas, manejadores de bases de datos, y lenguajes que se pueden usar de forma libre.

GLOSARIO

Archivo: Es un conjunto de información que se almacena en una computadora y puede ser identificado por su ruta completa.

Comando: Es una instrucción o mandato que el usuario proporciona al sistema, desde la línea de órdenes o una llamada a programa, el cual generalmente está contenido en un archivo ejecutable.

Consola: Se refiere a un terminal (generalmente una pantalla y un teclado) conectado a un computador, con acceso total y privilegiado al sistema.

Directorio: Agrupación de archivos de datos, atendiendo a su contenido, a su propósito o a cualquier criterio que decida el usuario. Técnicamente el directorio almacena información acerca de los archivos que contiene: como los atributos del archivo o dónde se encuentra físicamente en el dispositivo de almacenamiento.

Disco duro: Dispositivo encargado de almacenar información de forma permanente en una computadora.

Distribución: Es un conjunto de aplicaciones reunidas que permiten brindar mejoras para instalar fácilmente un sistema Linux (también llamado GNU/Linux).

Driver: Es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz -posiblemente estandarizada- para usarlo.

Emacs: Es un editor de texto altamente extensible y configurable creado por Richard Stallman, distribuido bajo la licencia libre GPL. En la actualidad es mantenido por la Free Software Foundation. Forma parte del proyecto GNU.

Ensamblador: Se refiere a un tipo de programa informático que se encarga de traducir un fichero fuente escrito en un lenguaje ensamblador, a un fichero objeto que contiene código máquina, ejecutable directamente por la máquina para la que se ha generado.

Ethernet: Es el nombre de una tecnología de redes de computadoras de área local (LANs) basada en tramas de datos. El nombre viene del concepto físico de ether. Ethernet define las características de cableado y señalización de nivel físico y los formatos de trama del nivel de enlace de datos del modelo OSI.

GNU: Es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

HTML: es un lenguaje para el marcado de textos e hipertextos.

HTTP – Hypertext Transfer Protocol.

Interfaz: Es la parte del programa informático que permite el flujo de información entre varias aplicaciones o entre el propio programa y el usuario.

Intérprete: Un intérprete es un programa capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel.

Kernel: Es el software responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma más básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

LAN: Es la abreviatura de Local Area Network (Red de Área Local o simplemente Red Local). Una red local es la interconexión de varias computadoras y periféricos. Su extensión esta limitada físicamente a un edificio o a un entorno de unos pocos kilómetros.

Las rutas absolutas: Son aquellas que se construyen poniendo toda la ruta hasta el directorio que nos queremos mover.

Las rutas relativas: Son aquellas que se construyen a partir del directorio en el que nos encontramos.

Liga: Es una referencia en un documento de hipertexto a otro documento o recurso.

Multiplataforma: Funciona con la mayoría de plataformas del mercado: Intel 386/486/Pentium, Motorola 680, Sun Sparc, etcétera.

Multiproceso: Permite la ejecución de varias aplicaciones simultáneamente.

Multiusuario: Distintos usuarios pueden acceder a los recursos del sistema simultáneamente aunque se trate de una instalación en una sola máquina.

Novell: Es una compañía de origen estadounidense dedicada al software, específicamente en el área de sistemas operativos de redes, como Novell Netware y Linux, entre otras ramas de la tecnología.

OpenOffice.org: Es una suite ofimática de software libre y código abierto que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos.

Particionamiento: Es la creación de divisiones lógicas en un disco duro que permite aplicar el formato lógico de un sistema operativo específico.

Password: Es una forma de autenticación que utiliza información secreta para controlar el acceso hacia algún recurso.

PHP: significa Hypertext Preprocessor.

Root: Es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (mono o multi usuario)

Shell: Es un programa informático lector de líneas de texto que un usuario de un ordenador ha predefinido y este programa lo interpreta para un sistema operativo o lenguaje de programación.

SQL, "Structured Query Language" representa un método estricto y más general de almacenamiento de datos que estándares anteriores.

StarOffice: Es la suite ofimática de Sun Microsystems, basado en el desarrollo Open Source del OpenOffice.org (que es patrocinado por la misma Sun). Está compuesto por un procesador de textos (Writer), planilla de cálculo (Calc), programa de manejo de base de datos (Base), de presentaciones (Impress), y de gráficos (Draw).

Swap: memoria virtual de Linux

Terabyte: Es una unidad de medida informática cuyo símbolo es el TB, y es equivalente a 2 a la 40 bytes.

UNIX: Es un sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

WWW: es llamado a un servidor que ofrece servicios dentro del Word Wide Web en Internet, es un programa encargado de ofrecer comunicación mediante el protocolo http.

BIBLIOGRAFÍA

YARGER, Randy Jay
MYSQL-PHP and mSQL Oram.
California : O'Reilly,1999. 487p

GREENSPAN, Jay.
MySQL-PHP database
Applications. New York : M &
T Books, c2001.6596p.

WELLING, Luke.
Php and sql web
development. Indianapolis :
SAMS, 2001.867p

MAZLAKOWSKI, Mark.
Aprendiendo MySQL en 21
días, México: Pearson Educación,
2001.534p.

BILKEN, Petra
PHP y MySQL : páginas web
dinámicas. Francia : KnowWare.
2002.80p.

ULLMAN, Larry
Guía de aprendizaje
MySQL. Madrid ; México:
Pearson Education: Prentice Hall,
C2003.328p.

BARKAKATI, Nabakakati.
Red hat Linux a fondo. Madrid:
Anaya multimedia, 1999. 780 p.

BLANCO, Vicente J.
Linux: Instalación, administración
y uso del sistema. México: Alfaomega,
C 1997. 788 p.

CARLING, M.
Guía avanzada administración de
Sistemas Linux. Madrid: Prentice Hall, 2000.
327 p.

DANESH. Armand.
La Biblia red hat linux 6.
Madrid: Anaya, 2000. 895 p.

LEBLANC, Dee-Ann.
Administración de sistemas Linux.
Madrid: Anaya, 2001. 864 p.

Edición especial Linux máxima
Seguridad. México: Prentice hall,
1999. 482 p.

ESPINOSA, Juan Carlos.
Red hat Linux 7.0 instalación y
Configuración básica. México:
Alfaomega, 2001. 192 p.

FACUNDO ARENA, Héctor.
Linux avanzado guía del
Administrador. Argentina: MP Ediciones,
2000. 279 p.

JUNCAR, José Antonio.
Todo sobre Linux red hat 6.1.Ra-B65418
Ma, 2000. 176 p

Páginas Web

1.-
http://groupsbeta.google.com/group/comp.os.minix/msg/b813d52cbc5a044b?hl=en

2.- http://www.gnu.org/philosophy/philosophy.es.html

3.- http://hotwired.goo.ne.jp/matrix/9709/5_linus.html

4.- http://www.gnu.org/philosophy/philosophy.es.html#AboutFreeSoftware