INTRODUCCIÓN AL SISTEMA OPERATIVO LINUX

Más de 50 Herramientas

MANUAL DE CAPACITACIÓN PARA EL TRABAJO

:>ls :>chmod :>cat -n :>vi :>grep :>cut :>find :>rm

1a. Edición

Daniel E. Vázquez Solís

INTRODUCCIÓN AL SISTEMA OPERATIVO LINUX

INTRODUCCIÓN AL SISTEMA OPERATIVO LINUX

DANIEL E. VÁZQUEZ SOLÍS.
Profesor del Departamento de Sistemas y Computación.
Instituto Tecnológico de Acapulco.

Geniux Press.
Editor.
www.geniux-online.com

ISBN 978-970-95789-0-4

Impreso en Servicios Gráficos Digitales Geniux Press.
Impreso en México.

Primera edición 2009.

AGRADECIMIENTOS

Esta obra tiene su origen en el año 2002 y desde entonces ha crecido y madurado gracias a las aportaciones y recomendaciones de muchos a quienes quisiera aprovechar para darles las gracias.

Quisiera expresar mi reconocimiento y en orden alfabético por primer nombre a: Adalid Dircio Benavidez, Adriana Nuñez Fierro, Carlos Ortíz Ortíz (q.d.e.p), Hernan Corrales Falcón, Jesús Ariel Gonzalez Espíritu, Oscar Arzeta Armenta y Rolando de la Mora por las contribuciones y sugerencias.

A todos ellos muchas gracias.

PRÓLOGO

Linux ha tenido un acelerado proceso de aceptación en la industria principalmente desde el año 2004.

Sin embargo siguen siendo las empresas grandes, medianas y gobierno quienes incluyen en sus estrategias el uso de Linux como herramienta para el desarrollo de parte de su infraestructura informática.

También es cada vez mas común que en los círculos de profesionales en informática y computación se hable acerca de Linux, inclusive para el ciudadano común es un término que empieza a dejar de ser ajeno.

A pesar de ser cada vez más conocido, aún no logra penetrar en las micro empresas o ser utilizado por los ciudadanos comunes con la misma velocidad con la que es conocido. ¿Porqué?

De mi experiencia como consultor y académico puedo dar varias razones, entre las que quiero destacar las siguientes dos:

a) La falta de entendimiento real de lo que es y no es Software Libre. Muchas veces aquellos que promovemos su uso utilizamos con ventaja el término "gratis" para recalcar su bajo costo de adquisición y con ello poder acercarlo al usuarios, sin embargo este término en muchas ocasiones no convence , genera desconfianza y la gente cree que hay "gato encerrado" por aquello que dice que "lo barato sale caro".

Por ello he incluido en el primer capítulo de este libro un conjunto de conceptos que ilustran lo que es realmente es el Software Libre, el Open Source y cuáles son sus mecanismos mas populares de comercialización.

b) Otro problema que impide el uso estandarizado de Linux, es la falta de personal calificado y que entienda realmente lo que significa usar Linux, es común que aprendamos a usar Linux sin dejar de pensar como usuarios de sistemas comerciales y esto genera conflictos debido a que no estamos hablando de lo mismo. Es por ello que este documento explica el uso de Linux desde su manejo en consola, a fin de que el lector comprenda la esencia de este sistema.

Es cierto que el uso de la consola de texto pudiera parecer arcaico, pero se dará cuenta que al dominarla entenderá cómo pensaban los antiguos hackers de las épocas, cuando la consola de texto estaba en sus momentos de gloria.

Este documento fue diseñado como un cuaderno de auto aprendizaje debido a que no es posible aprender Linux leyendo, hay que usarlo y sentirlo para conocerlo.

Espero sinceramente que este libro sea de su agrado y con la finalidad de mejorarlo dejo un correo electrónico para cualquier sugerencia o aclaración.

Por último, Para aquellos que sean conocedores de Linux y el Software Libre, pido una disculpa por utilizar Linux en lugar de GNU/Linux en el título de esta obra.

Daniel Enrique Vázquez Solís.
geniux_aca@yahoo.com.mx

CÓMO USAR ESTE LIBRO.

Esta obra es un manual de auto capacitación y por lo tanto le recomiendo al lector leerlo de manera continua desde el capítulo dos hasta el final, debido a que durante los ejercicios se hace referencia a comandos vistos anteriorente o se reutilizan archivos o directorios creados con anticipación.

En el recorrido de de la obra encontrará varios tipos de recuadros.

Nota:

Los recuadros con marco en doble raya hacen la aclaración de recursos que deben descargarse de Internet como elementos de apoyo a los ejercicios.

Los recuadros en fondo gris hacen referencia a lo que usted debe ver en la consola de texto.

El recuadro en blanco representa el contenido de un archivo de texto o se utiliza para resaltar las opciones de un comando.

Índice de contenido

SECCIÓN 1
EL SOFTWARE LIBRE.

1.1 INTRODUCCIÓN AL SOFTWARE LIBRE.

El Software Libre es un concepto cada vez es mas difundido entre informáticos, administradores, medios de comunicación e incluso usuarios comunes de servicios informáticos, sin embargo, a pesar de la difusión que ha tenido desde 2004, año de su despegue comercial, la mayoría de la gente tiene una idea muy vaga o no comprende realmente a que nos referimos cuando escuchamos las palabras Software Libre.

En esta sección, vamos a contestar algunas de las preguntas más comunes que surgen al escuchar de este tema por primera vez.

Tal vez usted lector se pudiera preguntar lo siguiente:

- ¿Qué es el Software Libre?
- ¿Qué es y como se interpreta la licencia de GPL del Software Libre?
- ¿Cómo se desarrolla el Software Libre?

En general, me interesa que el lector tenga un panorama un poco mas amplio de lo que realmente es el Software Libre, debido a que somos consumidores de este tipo de aplicaciones, pero no sabemos realmente por ejemplo, cuándo estamos utilizando realmente una aplicación desarrollada bajo este esquema.

1.2 ¿QUE ES EL SOFTWARE LIBRE?

Nos acostumbraron desde los años 80, que al adquirir una copia de software, el desarrollador de dicho software establece las condiciones de uso, prohibiendo entre otras cosas a quien lo adquiere, pueda distribuirlo entre familiares y amigos, tampoco nos permite a que alguien distinto al desarrollador pueda adaptarlo a sus necesidades, y ni pensar si quiera el poder corregir errores en caso de presentarse alguno.

Esto nos pudiera parecer algo normal, sin embargo vamos a ver el siguiente ejemplo:

Usted adquiere un cd o dvd con su software favorito y usted es dueño de 2 equipos de cómputo, pero la licencia dice que solo tiene permiso de instalarlo en un solo equipo. ¿Lo considera justo?

Obviamente que a la mayoría de la gente nos parece ilógico, debido a que físicamente el disco puede utilizarse en mas de un equipo sin sufrir daño.

Entonces, ¿porque me prohíben hacer algo que físicamente no representa por ejemplo un daño al equipo o al disco que en teoría es de mi propiedad, puesto que pague por el ?

En pocas palabras el fabricante nos restringe de manera legal (artificial) lo que podemos hacer con la copia que hemos adquirido y esto genera fenómenos tan penosos como la piratería de software.

Debido a lo aparentemente ilógicos que resultan estos esquemas de comercialización de software y a sus altos costos, hace algunos años, mas bien ya hace algunas décadas, un grupo de personas intentaron desarrollar un nuevo esquema de distribución de software, dando como origen la filosofía del Software Libre.

El Software Libre es el software que se desarrolla como cualquier otro programa pero se distribuye mediante un esquema orientado a la libertad.

El Software Libre tiene detrás de sí una filosofía que le garantiza al usuario 4 libertades básicas:

1. Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre.
2. Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente.
3. Libertad de redistribución, de modo que se nos permita colaborar con vecinos y amigos.
4. Libertad para mejorar el programa y publicar las mejoras. También exige el código fuente.

Esto quiere decir que un programa, para que pueda caer dentro de la definición de Software Libre debe distribuirse con su código fuente y no debe tener restricciones de uso.

De pronto, esto puede parecer una locura, pero en realidad este esquema de las 4 libertades básicas son únicamente la punta de un iceberg mucho mas complejo, que descubriremos gradualmente a lo largo de este capítulo.

En la práctica, para garantizar que nuestros programas no corran el riesgo de que los derechos de autor sean registrados a nombre de terceros, la gente que encabeza este movimiento, nos proporcionan también herramientas legales que le permitan a un software ser distribuido mediante este esquema de sin que el autor pierda los derechos sobre su obra.

La herramienta legal a la que nos referimos se le conoce como licencia GPL y se tocará mas a fondo en el apartado 1.6. Esta licencia describe las libertades bajo las cuales se distribuye un software, pero a su vez, establece las restricciones que impiden que terceros puedan apropiarse de las ideas del autor. Esto quiere decir que el autor conserva sus créditos como tal y establece las normas de su distribución mediante las 4 libertades básicas mencionadas anteriormente . Ademas nos obliga a que las mejoras hechas a a partir de un Software Libre también se distribuyan mediante la licencia GPL, contribuyendo de esta manera a la creación de mas Software Libre.

1.3 TÉRMINOS RELACIONADOS

Tal y como se menciono anteriormente, todas las aplicaciones de software son código que se ejecutan dentro de una computadora, sin embargo el debate a lo largo de este capítulo son los esquemas de licenciamiento bajo los cuales cualquiera de nosotros puede hacer uso de un programa.

La lista de conceptos que se presentan a continuación describen algunos de estos esquemas.

Open Source: Si usted lector, decide profundizar mas acerca del software libre y similares, es posible que se encuentre por ejemplo con el término *Open Source Software,* programas de fuente abierta. Los programas desarrollados mediante el esquema de fuente abierta, se distribuyen en un esquema que se centra mas en entregar el código fuente del programa y no tanto en las libertades del usuario.

Las aplicaciones de fuente abierta se centran en entregar el código fuente, sin embargo permiten restringir lo que se pueden hacer con este código . Estas restricciones pueden ir desde no establecer restricciones lo que lo hace idéntico al Software Libre, hasta impedir la distribución de código modificado, permitiendo la modificación solo al momento de la compilación.

Para que un programa sea considerada de fuente abierta su licencia debe cumplir con los siguientes lineamientos:

1.- *Libre distribución:* La licencia no debe obligarnos a pagar por la redistribución de un programa pero tampoco nos obliga a no recibir una remuneración por la distribución por un medio físico.

2.- *Acceso al código:* La aplicación pueden distribuirse tanto en código fuente así como código binario (compilado), pero cuidando que se tenga siempre acceso al código fuente del programa.

3.- *Trabajos derivados:* La licencia debe permitir la modificación del código y que los trabajos derivados sean distribuidos bajo la misma licencia.

4.-*Conservar la integridad del código fuente:* Las aplicaciones pueden ser distribuidas mediante los siguientes esquemas:

a) Con parches que permitan la modificación del código original a la hora de la compilación.
b) La licencia puede permitir la distribución modificada del código original.
c) La licencia puede exigir que los trabajos derivados tengan un nombre distinto.

5.- *No permitir la discriminación en contra de personas o grupos.*

6.- *No debe restringir su uso para alguna actividad humana en especial.*

7.- *Distribución de la licencia:* La licencia que acompañe a un programa de fuente abierta no debe requerir la incorporación de otra licencia en cualquiera de sus partes.

8.- *La licencia no debe ser especifica para un producto:* La licencia no debe requerir que para tener efecto sobre el software, este debe ser acompañado por algún otro producto específico.

9.- *La licencia no debe restringir otro software:* La licencia no debe tratar de extenderse a cualquier otro software o condicionar su uso solo en combinación con aplicaciones que tengan licencias similares.

10.- *La licencia debe ser tecnológicamente neutral:* Esto significa que la licencia no debe hacer referencia hacia alguna tecnología o interfaz en específico.

Esto nos genera un esquema similar al software libre sin llegar a la libertad total que nos ofrece la licencia GNU.

Si está interesado el conocer mas sobre este tema, consulte la referencia 4 que se presenta al final de este capítulo.

Freeware. Este término se refiere a programas gratuitos. Son binarios que se pueden redistribuir libremente pero no se proporciona el código fuente.

Shareware. No es software gratuito. Es un método comercial que se utiliza para distribuir programas y copiarlos libremente, pero no se pueden usar continuamente sin pagarlos. Después de cierto tiempo exige un pago para poder continuar haciendo uso de ellos.

Charityware o Careware. Por lo regular es software que se distribuye mediante un esquema similar al shareware, aquí el pago por su uso es una sugerencia y el dinero recaudado son para organizaciones caritativas.

Dominio público. Se refieren a programas en que el autor renuncia totalmente a sus derechos.

Copyleft. Este termino fue inventado por la gente que desarrolló la filosofía del Software Libre y hace referencia a un término opuesto al copyright, es decir el autor renuncia a exigir pago alguno por el uso de terceros del software, pero no renuncia a sus derechos de autor.

Propietario, cerrado, no libre. Es todo el software que no se considera libre ni de fuente abierta y que exige pago por su uso.

1.4 MODELOS DE NEGOCIO DEL SOFTWARE LIBRE.

El Software Libre no nació como un modelo de negocio, lo que busca este movimiento es que cualquiera tenga la libertad de ejecutar cualquier software para resolver sus problemas o satisfacer sus necesidades sin restricciones .

Recuerde: Es un asunto de libertad y no de precio.

Sin embargo, dadas las libertades que ofrece no le fue posible mantenerse fuera del ámbito comercial y se ha convertido en toda una industria paralela y/o complementaria al software comercial o de pago por licencia.

Desde mi muy particular punto de vista, el software libre es una industria de conocimiento en donde lo que se vende no es una licencia si no por el contrario se recibe una remuneración por satisfacer una necesidad.

De ahí que han surgido los siguientes modelos de negocio:

a) Uno de los grandes negocios es el soporte y esto tiene su origen entre otros en el hecho que, aunque se distribuya el software con sus respectivos manuales, el mercado exige a las empresas mejorar su rentabilidad, pero adaptándose a los cambios lo mas rápido posible, es por ello que no esperan a que su personal estudie un manual por largo tiempo y por ello terminan contratando expertos en el uso de esta aplicaciones libres .

b) El desarrollo de aplicaciones libres por si solas no parece un buen negocio, pero complementando esto con el soporte y/o la creación de fundaciones que permitan captar ingresos del gobierno y particulares por el mantenimiento de dicha aplicación puede permitir a un individuo por lo menos vivir holgadamente.

   Para que este modelo funcione se debe de tener una buena idea.

   Hablar de una buena idea puede ser por ejemplo desarrollar una aplicación que pueda resolver un problema muy común o el desarrollo de un reemplazo libre de una aplicación vendida bajo el esquema de pago por licencia y que sea muy popular, etc.

c) La industria del hardware puede ser un buen complemento al Software Libre debido a que a estas empresas les interesa vender sus equipos, pero para ello necesitan también disponer de software cuyo desarrollo sea continuo y económico. Si estas empresas encuentran una aplicación libre que pueda satisfacer sus necesidades, también están interesadas en realizar los respectivos donativos para sostener el mantenimiento de la aplicación.

   Recuerde que el costo del software también impacta en el precio final del hardware que lo contiene.

   La industria del los servidores y de los celulares son ejemplos de mercados muy propicios para el consumo de Software Libre.

1.5 EL NACIMIENTO DEL SOFTWARE LIBRE.

De acuerdo con Richard Stallman, quien es considerado el representante moral del movimiento de *Software* Libre, el compartir *software,* es tan antiguo como las computadoras. En los años 70's lo más natural era que cualquier programador pudiera

pedir el código a otro sin importar si éste, se encontraba en otra universidad o compañía.

Si recordamos el origen del negocio de las computadoras, en un principio eran equipos de propósito general incompatibles entre si y que se vendían a costos muy elevados y los modelos nuevos de un fabricante eran incompatibles con sus modelos anteriores, En esos años el negocio era vender el equipo, sin embargo necesitaban software y las empresas que desarrollaban esas computadoras no podían cubrir las necesidades de todos sus clientes. Es por ello entregaban el equipo con los códigos fuentes y el cliente adaptaba el equipo a sus necesidades muy particulares.

Con el tiempo, las computadoras empezaron a disminuir de tamaño, sus costos disminuyeron, el mercado creció y se abrieron nuevas oportunidades de negocio. La llegada de la PC prácticamente estandarizó la industria de la computación y con ella el negocio cambio radicalmente.

Cuenta sr. Stallman que a principios de los 80's, el laboratorio donde trabajaba , adquirió una computadora con un sistema operativo propietario, de ahí se empezó a cuestionar si era válido que una compañía no le permitiera hacer tus propias correcciones, y tuvieras que esperar que ésta, como dueña de los derechos del *software,* lo hiciera por el.

En base a la antigua práctica de compartir software y con la finalidad de defender la libertad de acceso al software, en 1985 el Sr. Richard Stallman y colaboradores crearon la fundación para el Software Libre (Free Software Fundation) con la finalidad de crear un sistema operativo libre y no depender de las compañías que se habían apropiado del mercado de una manera mañosa y artificial.

La idea base del desarrollo fue el sistema operativo UNIX®, el sistema mas popular de la época. De ahí que se crearon las herramientas libres necesarias para manejar este sistema operativo.

Para principio de la década de los 90`s el proyecto estaba casi completo pero faltaba el componente principal : el kernel o el núcleo del sistema el cuál fue creado un poco después y de casualidad por una persona ajena a esta fundación.

1.6 LA LICENCIA GPL

Con el fin de proteger el trabajo de los programadores de las comunidades de Software Libre y evitar que las grandes compañías patenten o registren a su nombre el

trabajo realizado por estas comunidades, se creó la Licencia Pública General (GPL), la que a grandes rasgos establece lo siguiente:

- Garantiza el derecho de copia, tanto de la aplicación como de su código fuente, siempre y cuando se mantengan los derechos del autor original.
- Permite que el código de una aplicación pueda modificarse siempre que se avise el cambio y se mantengan los derechos sobre el autor original.
- Si comparte una aplicación ejecutable debe poner a disposición de los usuarios su código fuente y manuales de uso.
- Permite a terceros cobrar por el acto de transferir Software Libre a fin de cubrir los gastos de esta acción, además que permite también, recibir una percepción por asesoría en el uso de Software Libre.

En el apéndice A se muestra la licencia GPL original descargada desde el sitio oficial de GNU (*http://www.gnu.org/copyleft/gpl.html*).

Le recomiendo al lector, que cuando instale un software en su equipo de cómputo, verifique las condiciones de uso que exige la licencia y así conocer los derechos y restricciones que tiene sobre el mismo.

REFERENCIAS ADICIONALES SOBRE EL TEMA.

1) Jesús Gonzalez Barahona, Joaquín Seoano Pascual , Gregorio Robles. *Introducción al software libre.*
http://www.uoc.edu/masters/oficiales/master_oficial_software_libre/master_oficial_software_libre_materiales.htm

2) Página oficial de la Fundación para el software libre.
http://www.fsf.org/

3) Página oficial de la fundación para el software libre en America Latina.
http://www.fsfla.org/

4) Página oficial de la Open Source Initiative.
http://www.opensource.org/

5) Listado de licencias compatibles con los lineamientos de la Open Source Initiative.
http://www.opensource.org/licenses/alphabetical

6) Página oficial de la GNU en español.
http://www.gnu.org/home.es.html

SECCIÓN 2
LOS ORÍGENES DE LINUX.

2.1 EL NACIMIENTO DE UNIX®.

Debido a la gran influencia que ha tenido UNIX® en el desarrollo de la mayoría de los sistemas operativos modernos, se ha incluido en este documento un fragmento de los orígenes de este sistema operativo.

La historia de Unix®, comienza a finales de los años 60's, cuando los Laboratorios Bell de AT&T® y el fabricante de computadoras en aquel entonces GE (*General Electric®*), trabajaron sobre un sistema operativo experimental denominado MULTICS. Este sistema fue diseñado como sistema interactivo para la computadora GE 645. La idea era que este sistema permitiera compartir recursos y a la vez, proporcionar seguridad a los usuarios. El desarrollo sufrió muchos retrasos y las versiones resultaron lentas y con grandes necesidades de memoria. Por ello, tiempo después los Laboratorios Bell, abandonaron el proyecto.

En 1969, Ken Thomson, uno de los investigadores de los Laboratorios Bell, involucrado en el proyecto MULTICS, diseñó un juego para la computadora GE llamado Space–Travel, que simulaba el sistema solar y una nave espacial. Thomson, vio que el juego se ejecutaba a jirones sobre la máquina GE y resultaba muy costoso, aproximadamente 75 dólares por ejecución. Con la ayuda de Dennis Ritchie (quien implantó el lenguaje C por primera vez en una DEC-PDP-11), Thomson reescribió el juego para ejecutarse sobre una DEC PDP-7, utilizando la estructura de un sistema de archivos que habían diseñado Ritchie, Rud Canaday y el propio Thomson. Él y sus colegas, crearon un sistema operativo multitareas, incluyendo sistemas de archivos, un intérprete de órdenes y algunas utilidades para el PDP-7.

Puesto que el nuevo sistema operativo multitarea para el PDP-7, podía soportar 2 usuarios simultáneamente, con muy buen sentido de humor se le llamó: UNICS (Uniplexed Information and Computing System), y para 1970, se había adoptado el nombre definitivo de UNIX®.

2.2 UNIX® COMO UN ESTÁNDAR.

La palabra UNIX®, se utiliza actualmente para definir a todo aquel sistema operativo que cumple con una serie de características que comparten dichos sistemas, como son:

- Soporte a multitareas.
- Soporte a multiprocesamiento.
- Soporte de un juego común de utilidades.

Actualmente existen en el mercado, una gran variedad de sistemas operativos cuya filosofía base es el UNIX® y como ejemplos podemos mencionar los siguientes:

- **AIX (IBM).**
- **SOLARIS (*Sun Microsystems*).**
- **SCO (*Santacruz Operation*).**
- **HP-UX (*Hewlett Packard*).**
- **FREE BSD.**
- **IRIX (*Silicon Graphics*).**
- **LINUX (*Software* Libre).**

2.3 CARACTERÍSTICAS DE UNIX®.

Shells programables

El shell o interprete de comandos es el subsistema que permite al usuario enviar órdenes a la computadora desde el teclado.

GNU/Linux soporta una gran variedad de intérpretes de órdenes , que ofrecen ambientes de trabajo diferentes, aunque todos son capaces de ejecutar cualquier comando.

Como ejemplos de *shells* tenemos: *C Shell, Bourne Shell, Korn Shell,* etc.

El árbol de directorios:

En Linux, sistema de archivos posee una estructura jerárquica en forma de árbol invertido, la raíz del sistema se encuentra en la parte más alta de árbol, se presenta con el caracter '/' y se ramifica en directorios y archivos.

La siguiente figura ejemplifica este concepto:

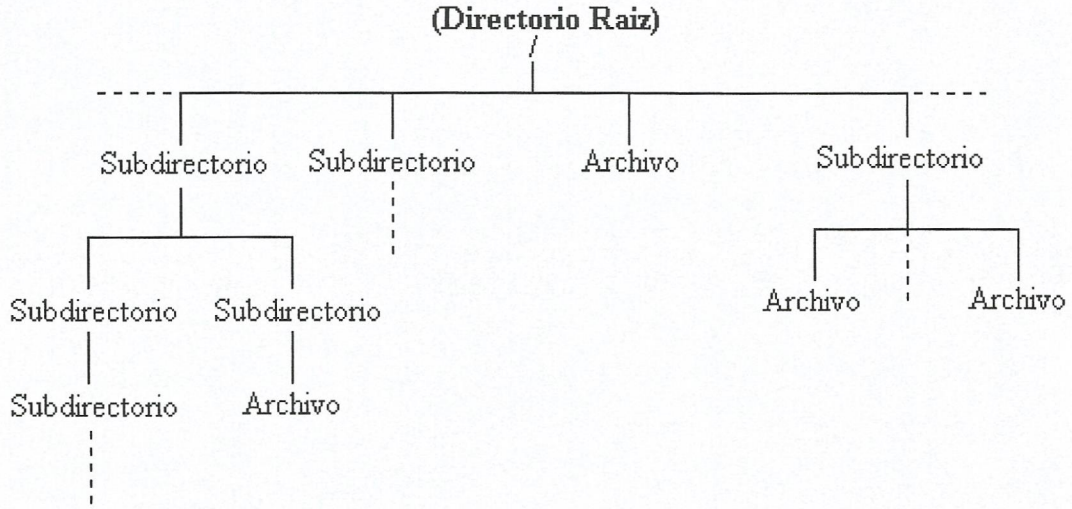


Figura 1. Árbol de directorios.

Este sistema reconoce una gran variedad de tipos de archivo, entre los que destacan:

- Archivos de sistema.
- Archivos de usuario.
- Archivos de dispositivo.
- Directorios.
- Ligas o Enlaces.

Mapeo de archivos hacia dispositivos:

Dentro de la filosofía de UNIX® y sus derivados, tenemos que los dispositivos están representados por archivos y es posible tener acceso por ejemplo a los periféricos, por medio de dichos archivos.

Caso especial son las unidades de almacenamiento secundario tales como disquetes, discos duros y unidades USB, las cuales además de estar asociados a un archivo de dispositivo, se relacionan con un directorio del sistema. Esto quiere decir que como parte del diseño de estos sistemas , aqui **NO** existen las unidades A:\, B:\ o C:\, todas las unidades de almacenamiento se asocian a un directorio del sistema de archivos por medio de una acción que se conoce popularmente como "montar" y una vez montado tenemos acceso a sus archivos.

Para ilustrar mejor estos concepto, a continuación mostraremos algunos ejemplos de archivos de dispositivos y el "dispositivo" que representan.

ARCHIVO	DISPOSITIVO
/dev/ttyS0	Puerto serial No 1
/dev/tty0	terminal de texto local No 0
/dev/pts/1	terminal de texto remota No 1
/dev/hda	Primera unidad de almacenamiento IDE
/dev/sda	Primera unidad de almacenamiento SCSI
/dev/cdrom	Unidad de CD-ROM
/dev/fd0	Unidad de almacenamiento extraible (floppy)

Organización del sistema:

El diseño general de este sistema operativo puede ser dividido básicamente en 3 capas las cuales podemos apreciar en la figura 2 :

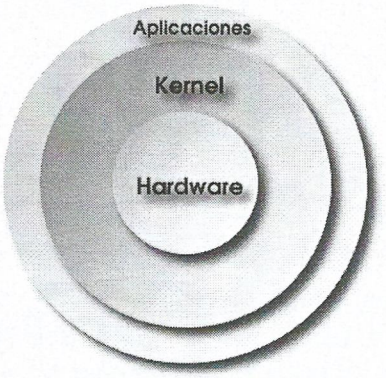


Figura 2. Estructura básica de un sistema basado en UNIX.

Al componente central se le conoce como *kernel* (o núcleo), sobre este kernel funcionan todas las aplicaciones, que hacen peticiones a dicho núcleo y así realizar sus operaciones.

El kernel es quien se comunica directamente con el harware.

Esta estructura permite el núcleo funcione como intermediario de comunicación entre las aplicaciones y el hardware, *lo que nos lleva a tener un mejor control y estabilidad en el sistema.*

2.4 EL NACIMIENTO DE GNU/LINUX.

Como se mencionaba en la sección 1.5, en 1990, Richard Stallman y su equipo tenian casí listo todo el juego de utilidades de un nuevo sistema operativo libre, sin embargo faltaba un componente: el kernel.

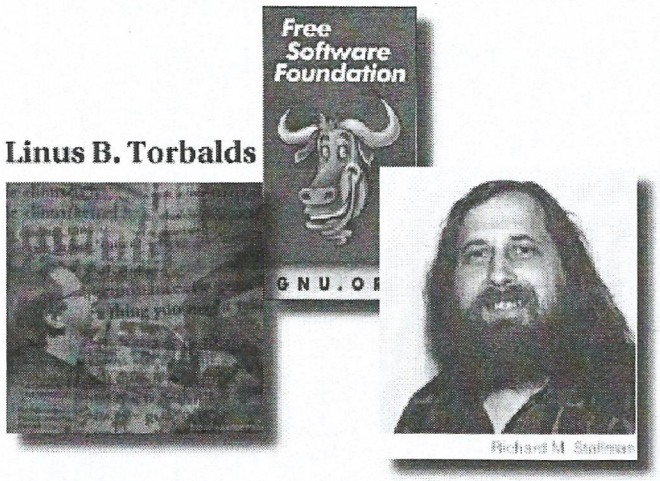


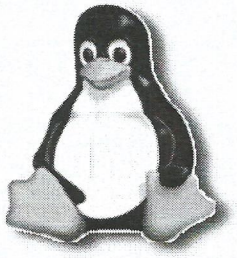
Figura 3. Precursores del software Libre y el Logo de GNU.

Afortunadamente un año más tarde, Linus B. Torvalds, por aquellos años estudiante de ciencias computacionales de la Universidad de Helsinki en Finlandia, buscando con afán también una alternativa para los sistemas operativos disponibles en el mercado, modificó el código del sistema operativo Minix, cuyo código fuente podemos encontrar en el libro de "Sistemas Operativos" escrito por el Dr. Andrew Stuart Tannebaum. De sus primeros avances los promocionó en el ciberespacio para verificar la bondades y errores del nuevo kernel.

El resultado fue la creación de una comunidad internacional de programadores que se unieron para el desarrollo de este nuevo sistema.

Ya cercano el año 1992, se logró unir el sistema GNU con el kernel de Linux y se obtuvo como resultado lo que conocemos hoy dia como el sistema GNU/Linux (que es en realidad como debe llamarse al sistema operativo Linux).

2.5 ¿QUÉ ES GNU/LINUX?



GNU/Linux, es un sistema operativo con las mismas prestaciones básicas que una versión comercial de UNIX ®.

En sus comienzos, esta plataforma estaba presente *casi* exclusivamente en el medio académico. Hoy, organizaciones científicas, organizaciones gubernamentales y otras grandes corporaciones, utilizan Linux como herramienta de trabajo.

Paralelamente académicos y programadores de todo el mundo auxilian su desarrollo, en conjunto con organizaciones científicas, empresas de software y gigantes de la informática como IBM® y HP®, entre otros.

Hoy en día, GNU/Linux posee diversas interfaces gráficas que facilitan su uso, inclusive para usuarios principiantes a quienes además, ofrece innumerables programas que componen un conjunto de aplicaciones de libre distribución. KDE y GNOME son los de escritorio que destacan por su sencillez visual y facilidad de uso.

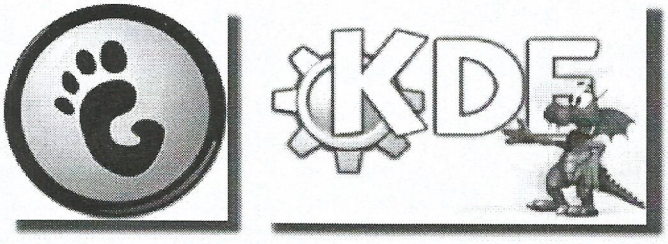


Figura 4. Logotipos de los dos grandes entornos gráficos en Linux.

Estas interfaces gráficas son reconocidas por su calidad de desarrollo, y esta característica hace que GNU/Linux, sea: El nuevo concepto en el mundo de la informática: Estable, rápido, código libre y abierto, con miles de programas y aplicaciones disponibles.

Linux, no sólo cuenta con un sin número de programas gráficos y de texto, sino que además posee aplicaciones para brindar soluciones a nivel empresarial ya sea de manera comercial o no comercial, entre ellas están suites de Oficina (Koffices, GNOffices, OpenOffices), manejadores de base de datos (PostgreSQL, MySQL, etc), servidores de correo electrónico (Sendmail, Qmail, Postfix, etc), servidores Web

(Apache), servidores de archivos (FTP, SMB, NFS, etc), entre otras muchas aplicaciones.

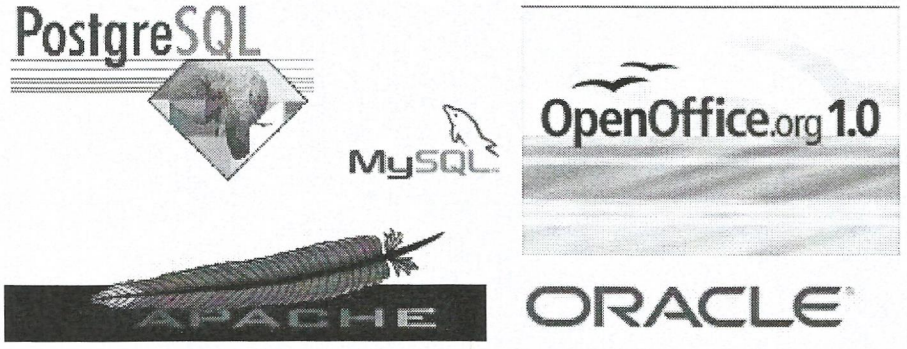


Figura 5. Algunas aplicaciones utilizadas en Linux.

Se reconocen más de cien distribuciones o marcas de GNU/Linux en el mundo, lo que en algunos casos provoca confusión entre los usuarios.

La diferencia fundamental entre un número de versión de Linux de cierta distribución y el número de versión de otra marca o distribución de Linux, es el conjunto de herramientas que ofrece y la versión del *kernel.*

Por desgracia, es muy difícil decir cuál es la mejor distribución o cuál es la versión de kernel más adecuada para un determinado problema. Actualmente la última versión del kernel de Linux, es la 2.6.x, en donde la 'x', representa a su vez la última "subversión" del kernel.

Recuerde que el kernel de Linux, siempre se encuentran en evolución y desarrollo.

Entre las distribuciones de Linux más populares tenemos:

1. Debian.
2. Red Hat.
3. S.U.S.E.
4. CentOS.
5. Mandriva.
6. LinEx.

7. Slackware.
8. Ubuntu.

Todas, y cada una de ellas, tienen sus diferencias, pero ninguna es completamente distinta a las demás, es decir, tienen puntos en común, como por ejemplo *el kernel*.

Nota: Todos los ejercicios de este libro fueron realizados utilizando la distribución Linux CentOS 5.2, con una versión de *kernel 2.6.18*.

Nota: Consulte la página http://www.geniux-online.com en donde encontrará información a cerca de los sitios en donde puede descargar esta distribución.

2.6 EL ENTORNO DE TRABAJO DE GNU/LINUX.

Como recordará, GNU/Linux es un sistema operativo multiusuario, esto significa que en el pueden convivir varios individuos (usuarios) trabajando sobre el mismo sistema de archivos y con la misma memoria de manera simultánea.

Para poder diferenciar cada uno de estos usuarios, y por seguridad, GNU/Linux exige que cada usuario tenga su propio identificador (o *login*), éste login es una cadena que puede contener caracteres alfa-numéricos y le ayudará a ingresar al sistema.

También es necesario dar de alta un *password (contraseña),* y se recomienda que tenga una longitud mínima de 8 caracteres y que sea absolutamente privado.

El *login* y el *password* es básicamente lo único que puede evitar que otros tengan acceso a sus archivos, así que tenga cuidado de que nadie mas lo conozca.

Una vez que un usuario ha sido dado de alta, el sistema operativo le asigna a cada usuario un directorio (carpeta) personal para que pueda manipular sus propios archivos. Por defecto ningún otro usuario tiene acceso a este directorio, con excepción del usuario administrador.

Este material ha sido diseñado para el aprendizaje de GNU/Linux desde la línea de comandos y utilizando un "interprete de comandos" en una consola de texto.

Nota: Consulte nuestra página http://www.geniux-online.com en donde podrá descargar una guía que le indica paso a paso como Instalar Linux CentOS 5.2 y así poder desarrollar los ejercicios contenidos en este documento.

Un intérprete de comandos (*shell*) está compuesto por un indicador (cursor), que espera una orden por parte del usuario.

La figura siguiente, muestra un ejemplo de un indicador en el intérprete de comandos.

```
[root@localhost ~]# _
```

Cuando aparece este 'indicador' (o *prompt* en inglés), es señal de que está esperando a que el usuario escriba un comando (mandato).

Al trabajar con Linux recuerde lo siguiente:

- Los comandos o mandatos de GNU/LINUX, siempre se escriben con letra minúscula.
- Los nombres de los archivos pueden tener una longitud máxima de 255 caracteres, y no existe el concepto de extensión, esto quiere decir que en GNU/Linux no existe el concepto de archivo .exe, .com, .bat, etc., y los archivos ejecutables pueden tener cualquier extensión o simplemente no tenerla; también los directorios pueden o no, tener extensión.
- Cuando asigne un nombre a un archivo o directorio, éste no debe de tener espacios en blanco.
- Cuando escriba un comando, separe el comando de sus argumentos con espacios en blanco.

2.6.1 COMANDOS Y MANDATOS.

Cuando nos iniciamos en el mundo de GNU/Linux es fácil encontrar que muchos usuarios están familiarizados con sistemas operativos con interfaz gráfica, y por lo tanto se muestren renuentes ante el uso de un entorno en texto, pero no olvide que los entornos de texto nos ofrece rapidez, eficiencia y facilidad en las operaciones.

Por esto:

"**La fuerza de Linux, reside en el manejo de su línea de comandos**".

Los comandos de texto en GNU/Linux, son mucho mas que eso, son autenticas herramientas que nos permiten realizar mas operaciones que los mismos ambientes gráficos.

Espero que al finalizar de leer este libro el lector logre sentir esa sensación de libertad que ofrece el dominio de las herramientas de texto.

SECCIÓN 3
LAS HERRAMIENTAS DE LINUX.

Como se mencionó en la sección anterior, es importante conocer el conjunto de herramientas que nos permitan realizar las distintas operaciones con un mínimo de esfuerzo.

Por ello en este capítulo, podrá encontrar las herramientas básicas para el manejo de la consola de texto divididas en las siguientes categorías:

- Herramientas básicas para manejo de archivos.
- Herramientas para manipulación de procesos.
- Herramientas para respaldo y recuperación.
- Herramientas básicas de red.

3.1 HERRAMIENTAS PARA EL MANEJO DE ARCHIVOS Y DIRECTORIOS.

Linux, es un sistema operativo con una estructura de directorios y archivos tan amplia, que puede intimidar a cualquiera que lo desconozca. Es como ir a la ciudad de México, por primera vez; lo primero que nos preguntamos al llegar a esta inmensa urbe es, cómo moverse y cómo localizar una dirección. Lo mismo sucede con el sistema de archivos: ¿Cómo puedo desplazarme en el?, ¿Cómo le digo lo que quiero hacer?

Para contestar estas preguntas, mostraremos las siguientes herramientas.

3.1.1 LA HERRAMIENTA *ls*

Despliega información acerca de los archivos y directorios en una ruta específica, sino se especifica la ruta, la herramienta toma por defecto el directorio actual de trabajo.

Formato:	ls [opciones | grupo de opciones] [lista de archivos o directorios]	
Opciones:	-l	Despliega la información de manera larga o detallada.
	-a	Despliega aquellos archivos que son considerados ocultos.
	-i	Muestra la información de cada uno de los i-nodos de un archivo.
	-1	Despliega en formato de una sola columna.
	-R	Si está indicado un directorio en la sintaxis, esta opción mostrará de manera recursiva el contenido del mismo.
	-d	Al indicar un directorio en la ruta, éste mostrará sólo la información del directorio indicado, mas no su contenido.

Esta herramienta es toda una tradición en UNIX®. Difícilmente, un sistema que diga derivarse de *UNIX®*, puede omitir el comando *ls*.

Antes de continuar con los comandos y si es la primera vez que utiliza una distribución GNU/Linux es importante tomar en cuenta la siguiente recomendación:

Para ingresar al sistema,escriba su nombre de usuario (*login*) y su contraseña (*password*).

Una vez adentro del sistema, observará algo como esto:

```
CentOS release 5.2 (Final)
Kernel 2.6.18.1 on an i686

localhost login: root
Password:
Last login: Sat May 16 12:08:44 on :0
[usuario@geniux ~]$ _
```

Recuerde que a esto se le conoce *prompt* (indicador), y especifica que a partir de ese momento es posible introducir comandos o herramientas, pero se preguntará, ¿cómo se interpreta la información anterior?

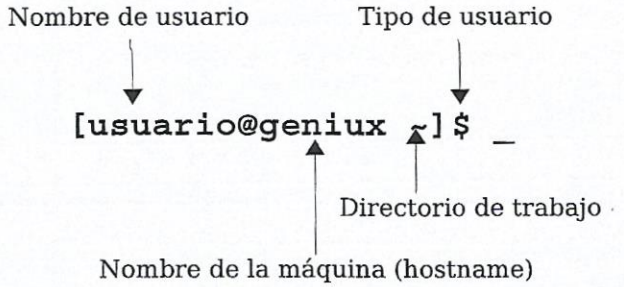


Figura 6. Muestra de manera descriptiva cada una de las parte que conforma el indicador del sistema Linux.

La figura 6 muestra los elementos que componen el prompt en CentOS 5.2. Este puede variar según la distribución de GNU/Linux que utilice pero su utilidad es siempre la misma.

Ahora vamos a introducir nuestro primer comando escribiendo lo siguiente:

```
[usuario@geniux ~]$ ls
```

Como resultado de la ejecución del comando anterior, pudiera no mostrar nada, debido a que es la primera vez que ingresa al sistema y por lo tanto no hay archivos en el directorio .

Por otro lado,la herramienta *ls*, genera la percepción de que el directorio está vacío, sin embargo esto por lo general no es cierto y para comprobarlo basta con ejecutar lo siguiente:

```
[usuario@geniux usuario]$ ls -a
```

Entonces puede devolver algo similar a lo siguiente:

```
. .. .bash_history .bash_logout .bash_profile .bashrc .kde
```

Cuando los nombres de archivo, empiezan con '.' (punto), no es posible verlos al invocar el comando *ls* sin argumentos, es por ello que los archivos cuyo nombre empiezan de esta manera se les consideran "ocultos".

El hecho de saber realmente cuales archivos hay en un directorio, no es suficiente; lo que resulta muy común es querer conocer por ejemplo, el tamaño que ocupa en *bytes,* un archivo y para ello, basta con invocar la herramienta ls, con el siguiente argumento:

```
[user@geniux  user]$ ls  -l
```

La opción *–l,* permite visualizar las características de cada uno de los archivos que se encuentra en el directorio indicado, o en el directorio actual de trabajo.

Si consideramos que el directorio solo contiene archivos ocultos, el resultado de la ejecución anterior podría ser lo siguiente:

```
total 0
```

Como podemos observar nos da un resultado de 0 bytes puesto que no tenemos ningún archivo visible, ahora podríamos preguntar lo siguiente: ¿Qué pasó con los archivos ocultos?.

Para poder responder esta pregunta se puede utilizar lo que se conoce como *"grupo de opciones". P*ara poder aclarar este punto, ejecute lo siguiente en la línea de comandos:

```
[usuario@geniux  ~]$ ls -l -a
```

El resultado será una salida como se muestra a continuación:

```
total 32
drwx------  3 usuario usuario  4096 oct 31 18:06 .
drwxr-xr-x 5 root     root     4096 oct 31 18:05 ..
-rw-------  1 usuario usuario  33   oct 31 18:28 .bash_history
-rw-r--r--  1 usuario usuario  24   oct 31 18:05 .bash_logout
-rw-r--r--  1 usuario usuario  176  oct 31 18:05 .bash_profile
-rw-r--r--  1 usuario usuario  124  oct 31 18:05 .bashrc
drwxr-xr-x 3 usuario usuario  4096 oct 31 18:05 .kde
```

Al utilizar varias opciones, se mezclan para poder dar un resultado combinado y es por ello que el ejemplo anterior se utilizaron las opciones *-l* y *-a*. Esto quiere decir que se desean todos los atributos de todos los archivos (incluyendo los ocultos).

Como se ha escrito desde el inicio de este documento, trabajar desde la línea de comandos en texto nos permiten trabajar con la mayor eficiencia posible, es por ello que es posible agrupar las opciones en una sola y así lograr que se escriban con un mínimo de caracteres.

Para comprobarlo, escriba lo siguiente:

```
[usuario@geniux ~]$ ls -la
```

El resultado es similar a la ejecución anterior con las siguientes ventajas:

- Se escribe un mínimo de caracteres.
- Se agrupan las opciones para obtener un resultado mas adecuados a nuestras necesidades.

Antes de continuar es importante poder explicar la información que nos presenta la herramienta ls al invocarse en su forma larga (-l).

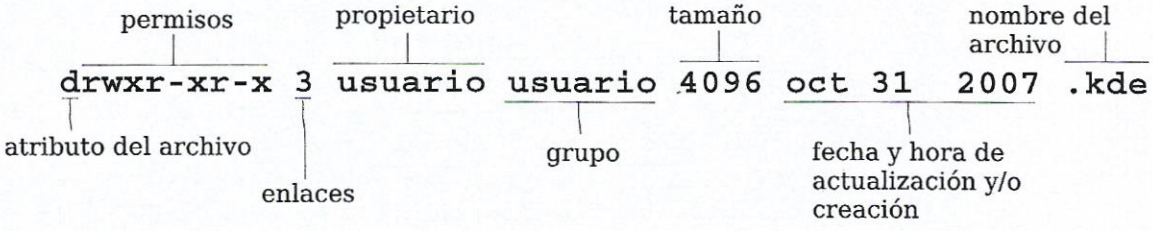


Figura 7. Atributos de un archivo.

La figura 7 muestra esta información y en los párrafos siguientes se explica cada uno de sus componentes:

- El primer carácter del primer campo (a la izquierda), denota el *"atributo de archivo"*, éste nos dice el tipo de archivo con el que se está tratando, y algunos de sus valores qposibles, son los siguientes:

  - *Archivo normal*
  - *d* *Directorios*
  - *c* *Archivo de dispositivo de carácter*
  - *b* *Archivo de dispositivo de bloque*
  - *p* *Archivo de dispositivo de tubería*

l *Archivo tipo liga*

s *Archivo tipo socket*

EJERCICIOS:

1.- Ejecutar las siguientes instrucciones y determinar el tipo de archivo que se trata.

```
[usuario@geniux ~]$ ls - l /dev/hda1
[usuario@geniux ~]$ ls - l /dev/tty1
[usuario@geniux ~]$ ls - l /dev/initctl
[usuario@geniux ~]$ ls - l /etc/rc.local
[usuario@geniux ~]$ ls - l /dev/gpmctl
```

- Dentro del primer campo, pero enseguida del tipo de archivo, tenemos los *"permisos"* y estos, se dividen en 3 bloques llamados bloque de *usuario, grupo y otros*, y cada bloque se compone de 3 letras:

r	*Permiso de Lectura*
w	*Permiso de Escritura*
x	*Permiso de Ejecución*

Esto significa que existen permisos de escritura, lectura y ejecución, para usuarios (propietario del archivo), el grupo (al que pertenece el propietario) y otros (cualquiera que no sea el usuario y no pertenezca al grupo).

La tabla siguiente muestra la forma como se distribuyen los permisos de un archivo.

R W X	R W X	R W X
Permisos de usuario	Permisos para grupo	Permisos para otros

- El siguiente campo representa el número de *links (ligas)*, esto se refiere a los enlaces que tengan con otros archivos o directorios. En el ejemplo de la figura 7, tenemos un directorio que contiene 3 enlaces hacia distintos archivos y subdirectorios.

- A continuación se presenta el nombre del *usuario propietario* del archivo.
- Después se puede observar el nombre del *grupo propietario del archivo.*

Hay que recordar que la mezcla de los campos de *propietario* y *grupo propietario,* en combinación con los derechos, determina quién tiene derechos de lectura, escritura y ejecución, sobre el archivo.

EJERCICIOS:

1.- De los siguientes archivos, determine: El tipo de archivo y los permisos que tienen el usuario propietario, el grupo propietario y otros.

- */etc/inittab*
- /bin
- /lib/libncurses.so.5
- /dev/ttyS0
- /usr/sbin/amrecover
- /dev/hda

Para finalizar, se pueden leer los últimos 3 campos que, por definición, se describen por sí solos:

- Tamaño del archivo expresado en bytes.
- Fecha y hora de creación o última modificación del archivo.
- Nombre del archivo.

3.1.2 LA HERRAMIENTA *touch*

Actualiza la fecha de modificación de un archivo siempre y cuando el archivo exista, si no es así, crea un archivo vacío.

Formato:	touch [opciones | grupo de opciones] lista-de archivos.	
Opciones:	-c	no crea archivo.
	-a	cambia la fecha de acceso del archivo.
	-m	cambia la fecha de modificación del archivo.
	-r	Cambia la fecha de un archivo tomando a otro como prototipo.

-t <fecha> Opción que permite determinar una fecha dada en: **YYYYMMDDhhmm**

La herramienta *"touch"*, se utiliza regularmente para crear archivos vacíos, siempre y cuando no exista un archivo o directorio con el mismo nombre. De ser así, simplemente actualiza la fecha de acceso.

Para comprobarlo basta con escribir lo siguiente:

```
[usuario@geniux ~]$ touch hola.c
```

Ahora utilice el comando *"ls"*, para verificar la fecha de creación y compruebe que su tamaño es de cero *bytes*.

EJERCICIOS:

1.- Determine que efecto tienen los siguientes comandos:

```
[usuario@geniux ~]$ touch -r .bashrc hola.c
[usuario@geniux ~]$ touch -t 200906011300 hola.c
```

3.1.3 LA HERRAMIENTA *rm*

Permite eliminar o borrar archivos y/o directorios. Estos últimos, deben tener permisos de lectura/escritura.

Formato:	rm [opcion | grupo de opciones] lista-de-archivos
Opciones:	-f Permite NO pedir confirmación a la hora de borrar archivos.
	-i Pide confirmación cada vez que se va a eliminar un archivo.
	-r Borra de manera recursiva un directorio y todo su contenido.

Antes de hacer cualquier prueba tenga cuidado al utilizar la opción – *f*, ésta, evita que el comando haga la pregunta de confirmación antes de borrar un archivo. En algunas distribuciones, esta opción esta activada por defecto, en estos caso se recomienda hacer uso de la opción *-i* .

NOTA: *Al eliminar un archivo desde la línea de comandos no es posible recuperarlo.*

EJEMPLOS:

Para poder iniciar las pruebas será necesario crear el siguiente conjunto de archivos:

```
hola
hola.z
hola.h
```

Ahora hay que suponer que debe borrar todos los archivos cuyo nombre comiencen con la palabra *"hola"* sin importar el resto de los caracteres.

Esto se logra de la siguiente forma:

```
[usuario@geniux ~]$ rm hola*
```

En el ejercicio anterior, el asterisco es conocido como *"comodín"* y sustituye a cero o más caracteres, después de la palabra *"hola"*.

En Linux, podemos encontrar diferentes tipos de comodines. El asterisco es uno de ellos, pero no el único, por lo que es posible utilizar otros más. En la siguiente tabla se muestran los comodines más comunes.

NOTA: L*os comodines funcionan con la mayoría, de los comandos del sistema, todo depende del contexto al que se aplican.*

Comodín	***Descripción:***	***Ejemplo***	***Podría ser ...***	***No es ...***
***	*Sustituye cero o mas caracteres.*	*hola**	*hola, holas, hola.z, holasis*	*hol, ho, h*
?	*Sustituye un solo carácter*	*hol?*	*Hola, hole, holi, holo, holu, ...*	*Hol, ...*
[<lista>]	*Sustituye cada miembro de la lista dada*	*[lct]oco*	*loco, coco, toco*	*Cualquier otra diferente.*
[inicial ... final]	*Sustituye cualquier miembro del rango dado.*	*[a-h]to*	*ato, bto, cto, dto, esto, fto, gto, hto*	*Cualquier otro diferente.*

Comodines mas usados en el sistema.

La filosofía a la hora de trabajar con esta herramienta es buscar un patrón que nos permita agrupar los elementos a eliminar.

Para realizar una segunda prueba, hay que crear los siguientes archivos:

```
pal45.c
cal56.g
sal88.h
```

¿Tienen algo en común?, ¿Cómo sería posible eliminar los 3 archivos, con una sola instrucción?.

Una solución posible es la siguiente:

```
[usuario@geniux  ~]$ rm ???[4-8][5-8].*
```

¿ Por qué ?

Vamos analizar la expresión anterior:

a) Los tres comodines ??? me dicen que busco nombres archivos que comiencen con 3 caracteres, los que sean.
b) El comodín [4-8] me indica que el nombre del archivo, después de los 3 ??? debe seguir un número entre 4 y 8.
c) El comodín [5-8] determina que siguiente símbolo en el nombre de archivo, debe ser un número comprendido entre 5 y 8.
d) El .* nos dice que el nombre de archivo debe tener un punto y terminar con cualquier secuencia de 0 o mas caracteres.

EJERCICIOS:

1.-Ahora observe el siguiente grupo de archivos.

```
pala
pico
cal2
```

Eliminar con una sola instrucción los archivos anteriores, suponiendo que existan más archivos que no deseo borrar?

2.-Para los siguientes archivos.

pato.h	lobo.txt
robo.pdf	bobo.ps

Eliminar únicamente *lobo.txt* y *bobo.ps* utilizando una sola instrucción.

3.- Crear los siguientes archivos en su directorio de trabajo.

pata	robalo.l
pato.h	robo.pdf
rabano.l	relax.c
rabo.j	

Eliminar los archivos *rabano.l, rabo.j, robalo.l, robo.pdf* utilizando una sola instrucción.

4.-Crea los siguientes archivos:

Zeferino.pdr	Pablo.pso
Zacarias.mnt	Yoli.aca
Alejandro.tld	pascual.bng
pep.si	coca.cola

Eliminar los archivos *Zeferino.pdr, Zacarias.mnt, Alejandro.tld, Pablo.pso,* utilizando un sola instrucción y sin borrar el resto de los archivos.

3.1.4 LA HERRAMIENTA *mkdir*

Crea uno o más directorios de trabajo

Formato:	mkdir [opciones] lista-de-directorios
Opciones:	-p Crea los directorios padre que falten para cada argumento directorio.

El comando *mkdir,* es una herramienta bastante sencilla que permite crear uno o varios directorios, de manera rápida y sin complicaciones. Para ilustrar su uso vamos

a suponer que se desea crear un directorio llamado prueba. Para ello, habría que utilizar la herramienta como se muestra en el recuadro siguiente:

```
[usuario@geniux ~]$ mkdir prueba
```

Ahora verifique con la instrucción *ls - l,* cómo aparece la letra *'d'* a la izquierda de los atributos:

La herramienta *mkdir*, es a los directorios, lo que la herramienta *touch,* para los archivos, crea generalmente directorios vacíos, aunque es posible crear directorios ya con subdirectorios en su interior, utilizando el parámetro -p. Para comprobarlo realice el siguiente ejercicio:

EJERCICIOS:

1.- Cree los siguientes directorios en su directorio de trabajo y compruébelo con el comando *ls –lR.*

```
prueba2
prueba3/bin
prueba4/etc/X11
```

3.1.5 LA HERRAMIENTA *tree*

Muestra en pantalla, en forma de árbol, todos los archivos y directorios de un directorio dado.

Formato:	tree [opciones] lista-de-directorios	
Opciones:	-a	Muestra todos los archivos y directorios incluyendo los archivos denominados ocultos.
	-d	Enlista sólo los directorios y sus subdirectorios.
	-f	Lista con la ruta completa de cada uno de los archivos y directorios implicados.
	-p	Muestra los atributos de los archivos/directorios
	-n	Quita los colores de los directorios.
	-u	Muestra el nombre del usuario propietario del archivo/directorio

```
-g  Muestra el nombre de grupo dueño del archivo/directorio
-s  Muestra el tamaño del archivo/directorio.
```

Esta herramienta tiene la capacidad de representar en forma de árbol y de forma gráfica, los archivos/directorios de un directorio dado. A continuación se sugieren algunos ejemplos para demostrar el uso de este comando:

EJEMPLOS:

Ejecutar los siguientes comandos sobre su directorio de trabajo para crear un pequeño árbol de directorios.

```
[usuario@geniux ~]$ mkdir -p dirA/dirB/dirC
[usuario@geniux ~]$ mkdir -p dirA/dirD/dirE/dirF
```

A continuación ejecutar el comando *tree,* sobre el directorio *dirA.*

```
[usuario@geniux ~]$ tree dirA
```

El resultado debe ser el siguiente:

```
dirA
|-- dirB
|   `-- dirC
`-- dirD
        `-- dirE
            `-- dirF

5 directories, 0 files
```

EJERCICIOS:

1.- Trabajando con la misma estructura, pruebe las siguientes instrucciones con las demás opciones, e intente deducir la salida de acuerdo con las descripciones del cuadro de opciones presentado al inicio de esta sección.

```
[usuario@geniux ~]$ tree -a dirA
[usuario@geniux ~]$ tree -d dirA
[usuario@geniux ~]$ tree -f dirA
[usuario@geniux ~]$ tree -n dirA
[usuario@geniux ~]$ tree -p dirA
```

Al terminar éstas, prueba sólo ejecutar el comando sin parámetros y observa lo que sucede.

3.1.6 LA HERRAMIENTA *rmdir*

Borra o elimina un directorio, siempre y cuando el directorio se encuentre vacío.

Formato:	rmdir lista-de-directorios
Opciones:	No hay opciones

Este comando por sí solo es de poca utilidad, debido a que NO se puede utilizar sobre directorios que NO estén vacíos.

EJEMPLOS:

Para comprobarlo basta con tratar de eliminar el directorio *prueba3*, creado anteriormente como se ilustra a continuación:

```
[usuario@geniux ~]$ rmdir prueba3
```

El resultado sería el siguiente:

```
rmdir: `prueba3': El directorio no está vacío
```

Para poder eliminarlo habría que seguir los siguientes pasos:

```
[usuario@geniux ~]$ rmdir prueba3/bin
[usuario@geniux ~]$ rmdir prueba3
```

La ilustración anterior muestra que para eliminar directorios NO vacíos con *rmdir*, primero hay que eliminar su contenido antes de poder borrar el directorio.

EJERCICIOS:

1.- Eliminar del directorio *dirA*, el subdirectorio *dirC y dirF.*
2.- Eliminar todos los directorios de *dirA, prueba2, prueba4, así como su contenido.*

3.1.7 LA HERRAMIENTA *cp*

Esta herramienta permite copiar uno o más archivos ordinarios, incluyendo archivos de texto y programas ejecutables. Tiene dos modos de operación: El primero hace una copia un archivo en otro; el segundo, copia uno o más archivos a un directorio.

Formato:	cp [opción | grupo de opciones] archivo-fuente archivo-destino. cp [opción | grupo de opciones] lista-de-archivos-fuente directorio-destino.	
Opciones:	-i	Permite advertir, en caso de que el archivo a copiar ya exista.
	-f	Desactiva la opción de advertir en el caso de sobreescribir.
	-p	Permite copiar un archivo con todos su atributos como son: los permisos, el propietario, el grupo propietario y su fecha .
	-r	Copia de manera recursiva (toda su información) un directorio.
	--parents	Copia un archivo creando toda la estructura de directorios por debajo de él.

EJEMPLOS:

Crear los siguientes archivos y directorios en su directorio de trabajo.

```
[usuario@geniux ~]$ touch archivoCP1 archivoCP2
[usuario@geniux ~]$ mkdir dirCP1
[usuario@geniux ~]$ mkdir -p dirCP2/dirCP3/dirCP4
[usuario@geniux ~]$ touch dirCP2/dirCP3/dirCP4/archivoCP4
```

+ A continuación intente copiar el contenido del archivo llamado *"archivoCP1"*, hacia el archivo *"archivoCP3"*. En el siguiente recuadro se muestra como hacerlo.

```
[usuario@geniux ~]$ cp archivoCP1 archivoCP3
```

La operación anterior crea una copia del archivo *"archivoCP1"*, en *"archivoCP3"*, sobre el directorio de trabajo.

+ Ahora, intente copiar de nuevo el archivo llamado *"archivoCP1"*, hacia el archivo *"archivoCP2"*, pero como se muestra a continuación y observe lo que sucede:

```
[usuario@geniux ~]$ cp -i archivoCP1 archivoCP2
```

Debe aparecer el siguiente mensaje:

```
cp: ¿sobreescribir «archivoCP2»? (s/n)
```

NOTA: *En algunas configuraciones de usuario, la opción "-f" se encuentra predeterminada, es por ello que de dicha opción no es mostrada en los ejemplos.*

La opción –i, tal como se muestra en el recuadro al inicio de esta sección, establece que esta herramienta nos pregunte si se desea sobreescribir el contenido del archivo. Utilícese esta opción, si el contenido de los archivos es importante.

Ahora hay que hacer una copia tal y como se muestra en el recuadro siguiente.

```
[usuario@geniux ~]$ cp --parents dirCP2/dirCP3/dirCP4/archivoCP4 dirCP1
```

Para comprobar lo sucedido se puede utilizar el comando *tree*, de la siguiente manera:

```
[usuario@geniux ~]$ tree dirCp1
```

La opción –parents copia toda la estructura seleccionada del dirCP2 adentro del directorio dirCP1.

- Copiar todos los archivos de directorio actual dentro del directorio *dirCP1.*

```
[usuario@geniux ~]$ cp * dirCP1
```

Con este comando, enviarás algunos errores, debido a que por defecto no copia directorios, pero para solucionar esto, se deberá utilizar el parámetro -r.

```
[usuario@geniux ~]# cp -r * dirCP1
```

cp permite copiar uno o varios archivos hacia otro directorio. Al utilizar este comando, es válido recurrir al uso de los comodines para facilitar su uso.

EJERCICIOS:

Para todo lo anterior, hay que crear en su directorio personal, la estructura de directorios y archivos que se muestran en la figura 8.

Ahora se utilizará la herramienta *cp*, aprovechando la estructura de la figura 8 en los siguientes ejercicios:

Recuerde que no debe moverse del directorio de personal de trabajo.

1. Copiar el archivo *swaps.swp*, hacia el directorio *bin*.
2. Copiar el archivo *swaps.swp*, hacia el directorio *etc.*, y llámelo *swapfile.swp*.
3. Copiar el archivo *teléfono.txt*, de *user1*, hacia *user2*, y llámelo *agenda.txt*.
4. Copiar *libdb.so*, hacia el directorio de trabajo del usuario (actual).

Nota: El archivo telefono.txt es un archivo que se utilizará a lo largo de este libro y debe descargarse del sitio http://www.geniux-online.com

RECOMENDACIONES: *Se debe crear la estructura de directorios sin salir de su directorio personal, para ello debe utilizar la opción "-p", al momento de crearlos, también, al crear los archivos debe utilizar el comando "touch" sin salir de su directorio personal.*

No descartar, ni eliminar la estructura anterior de archivos y directorios, al terminar el ejercicios debido a que será utilizada posteriormente.

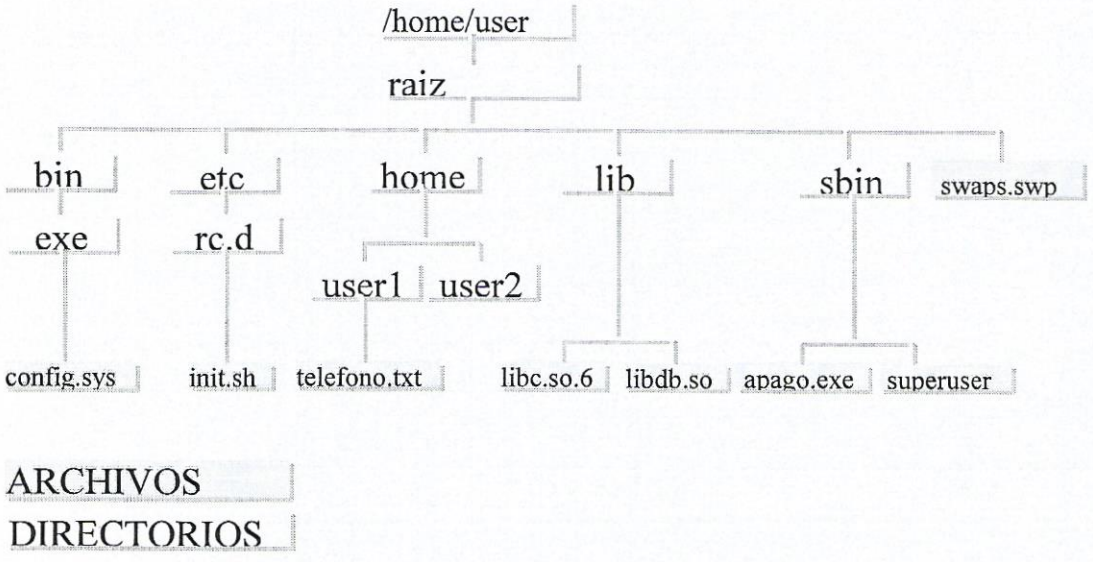


Figura 8. Estructura de directorios para el ejercicio.

3.1.8 LA HERRAMIENTA *mv*

Esta herramienta permite mover archivos o directorios hacia cualquier otro directorio, además que puede ser utilizado para renombrar archivos y/o directorios.

Formato:	mv [opción | grupo de opciones] archivo nombre-nuevo. mv [opción | grupo de opciones] lista-de-archivos directorio.
Opciones:	-f No pide confirmación de sobrescribir, si un archivo ya existe con el mismo nombre.
	-i Pide confirmación cuando existe el archivo destino.

La herramienta *mv*, puede ser utilizada de la misma forma que el comando *cp*.

Para ilustrarlo, supóngase que existe el archivo *hola.c*, y desea cambiar su nombre a *bola.h*. Para lograrlo, basta con escribr y ejecutar lo siguiente:

```
[usuario@geniux ~]$ mv hola.c bola.h
```

Con este comando también es posible renombrar un directorio como se ilustra en el siguiente ejemplo donde tenemos un directorio llamado originalmente *dir.h, y* que renombrará a *midir*

```
[usuario@geniux ~]$ mv dir.h midir
```

También es posible mover un directorio con todo y su contenido hacia otro directorio. Para ello supóngase que tenemos 2 directorios: uno llamado *uno.dic,* y otro *dos.doc.* Para mover el directorio *uno.dic,* hacia el interior de *dos.doc,* basta con ejecutar lo siguiente:

```
[usuario@geniux ~]$ mv uno.dic dos.doc
```

EJERCICIOS:

Utilizando la estructura de directorios y archivo de la figura 8, realice los siguientes ejercicios:

RECOMENDACIONES: *Realice los ejercicios sin moverse de su directorio de trabajo. Para verificar cada uno de los ejercicios propuestos utilice la herramienta "tree".*

1. Cambie el nombre del archivo *swaps.swp,* por el nombre de *intercambio.swp.*
2. Cambie el nombre del directorio *etc.,* por el de *etcétera.*
3. Mueva el archivo *config.sys,* hacia el directorio *etcétera.*
4. Mover todos los archivos de *lib,* (no mover el directorio) hacia el directorio *raíz.*
5. Mover todo el directorio *etcétera,* por debajo de *lib.*
6. Mover el directorio *sbin,* al directorio de trabajo del usuario.

3.1.9. LA HERRAMIENTA *cd*

Esta herramienta permite desplazarse por cada uno de los directorios que existen en el sistema de archivos.

Formato:	cd [directorio]
Opciones:	No hay opciones

Hasta ahora se ha insistido en permanecer en la raíz del directorio de trabajo. Esto no debe representar un inconveniente, aunque ya es tiempo de poder alcanzar mayor libertad.

A partir de ahora, será libre de poder desplazarse a cualquier punto del sistema de archivos.

Se comenzará por ver algunos ejemplos de cómo es posible aprovechar esta sencilla herramienta.

EJEMPLOS:

1.- Supongamos que desea posicionarse desde el directorio usuario el directorio llamado *raíz* de la figura 8. *P*ara ello, hay que escribir lo siguiente:

```
[usuario@geniux ~]$ cd raiz/
```

2.- Ahora supongamos que desde adentro del directorio raíz, desea moverse hacia el directorio *rc.d,* que se encuentra dentro del directorio etc en *raíz (ver figura 8).*

```
[usuario@geniux raiz]# cd etc/rc.d
```

3.- Estando en el directorio *rc.d,* desea moverse hacia el directorio *home* del directorio *raíz* (ver figura 8).

```
[usuario@geniux rc.d]# cd ../home
```

Las rutas que hemos manejado en los ejercicios anteriores de la herramienta cd , se llaman "*rutas relativas*" debido a que especificamos la ruta a partir del directorio actual.

Cuando especificamos una ruta a partir del directorio raíz (*/*) entonces se dice que estamos utilizando "*rutas absolutas*". Estas se manejan de la siguiente manera:

4.- Desde el directorio actual, pasar al directorio rc5.d del sistema:

```
[usuario@geniux  rc.d]# cd /etc/rc.d/rc5.d
```

5.- Cuando nos iniciamos en Linux, muchas veces entramos a los distintos subdirectorios del sistema por curiosidad y de ahí brincamos a otros, pero resulta que llega el momento en que estamos perdidos y no sabemos cómo regresar a nuestro directorio de inicio (directorio de usuario). Para estos casos basta con escribir lo siguiente:

```
[usuario@geniux  home]# cd
```

Al invocar la herramienta cd por si sola nos regresa a nuestro directorio de usuario.

EJERCICIOS:

Aprovechando la estructura de directorios de los ejercicios anteriores, compruebe sus habilidades.

1. Moverse hacia el directorio *exe.*
2. Estando en el directorio *exe,* moverse al directorio *sbin* dentro del directorio *raíz.*
3. Moverse del directorio sbin actual hasta el directorio */usr/local/bin.*
4. Dentro de ese directorio, irse al directorio principal del sistema (/) sin utilizar *cd /.*

Recuerde que en todo directorio, tenemos siempre por sistema, dos subdirectorios: El subdirectorio '.' (punto) y el subdirectorio '..' (punto, punto).

El subdirectorio '.' se refiere al directorio actual y el subdirectorio '..' apunta hacia el directorio inmediato superior.

3.1.10 LA HERRAMIENTA *pwd*

Muestra la ruta completa desde el directorio raíz, hasta el directorio actual .

Formato:	pwd
Opciones:	No hay opciones

Al invocar la herramienta *pwd* devuelve algo como esto:

```
[usuario@geniux ~]$ pwd
/home/geniux
```

El resultado del ejemplo anterior se interpreta que, quien invoca la herramienta pwd se encuentra trabajando en el directorio /home/geniux.

EJEMPLOS:

1.- Entrar al directorio *raíz/home/user1* y verifique con *pwd,* su ubicación.

```
[usuario@geniux ~]$ cd raiz/home/user1
[usuario@geniux user1]$ pwd
/home/usuario/raiz/home/user1
```

3.1.11 ENTRADA ESTÁNDAR, SALIDA ESTÁNDAR Y ERROR ESTÁNDAR.

Hasta este momento hemos visto cómo trabajar con archivos y directorios, pero aún nos es desconocido algún método que nos permita ver el contenido de un archivo o mejor aún, modificarlo.

La siguiente serie de comandos permiten ver hacia el interior de los archivos, y mostrar su contenido en pantalla, sin embargo, aprovecharemos esta sección para introducir 3 conceptos básicos de las consolas de texto:

- La entrada estándar.
- La salida estádar.
- El error estándar.

En este momento debe estar utilizando una terminal de texto y cada vez que ejecuta un comando envía el resultado a pantalla. Esto puede parecer muy lógico, pero no necesariamente tiene que ser así. Todas las herramientas de Linux, que devuelven un valor, lo envían hacia lo que se conoce como "**salida estándar**".

La salida estándar, por defecto es la pantalla (monitor), pero no es regla que siempre sea así. Un poco más adelante se explicará cómo se puede hacer modificaciones para que la salida sea dirigida hacia otro dispositivo.

En GNU/Linux, cuando una herramienta de texto emite un mensaje de error, éste, se envía hacia lo que se conoce como el "**error estándar**", que por defecto, también es la pantalla, que igualmente es susceptible de reenviarse hacia otro dispositivo.

Por último, en las terminales de texto tenemos también lo que se conoce como "**entrada estándar**", que es el dispositivo por el cuál se recibe información al sistema por defecto. La entrada estándar por defecto es el teclado.

Lo anterior lo podemos resumir de la siguiente manera:

ENTRADA ESTÁNDAR	Corresponde tradicionalmente al teclado y es el dispositivo por el cuál se reciben las órdenes por defecto.
SALIDA ESTÁNDAR	Corresponde al dispositivo hacia donde se envía por defecto la salida generada por una herramienta de texto y normalmente es el monitor.
ERROR ESTÁNDAR	Corresponde al dispositivo hacia donde se envían por defecto los mensajes de error, generados por las herramientas de texto y que tradicionalmente corresponden al monitor.

Hecha la aclaración se procederá al análisis de las siguientes herramientas:

3.1.12 LA HERRAMIENTA cat

Muestra en pantalla el contenido de un archivo de uno o más archivos.

Formato: cat [opciones] [lista-de-archivos]
Opciones: -n muestra el número de línea.

Para hacer pruebas con esta herramienta, es necesario disponer de un archivo que contenga información, así que, se hará uso de los archivos ocultos ¿Los recuerda?

Vamos a comenzar por echar un vistazo a nuestro archivo .bashrc, de la siguiente manera:

```
[usuario@geniux ~]$ cat .bashrc
```

Su contenido es extraño, pero por ahora sólo nos interesa poder ver qué hay en su interior. Hágase lo mismo con el resto de los archivos ocultos y observe su contenido.

Ahora se aprovechara este comando para sacar una copia del contenido del archivo *.bashrc,* sin utilizar el comando *cp,* y guardarlo en un archivo llamado *bashi.sh.*

Esto se logra de la siguiente manera.

```
[usuario@geniux ~]$ cat .bashrc > bashi.sh
```

El carácter '>', redirecciona la **salida estándar** hacia el archivo especificado a su derecha. Esto quiere decir que en lugar de enviar el contenido del archivo hacia el monitor, se reenvía al archivo *bashi.sh.*

Considérese ahora, escribir lo siguiente:

```
[usuario@geniux ~]$ cat > lomo
```

Después de ello, introdúzcase la siguiente información:

```
Nombre: Juan Peralta
Dirección: Calle de la Amargura #10
Edad: 25
Sexo: Masculino
```

Para cortar la secuencia y almacenar la información pulse *[ctrl + d],* y verifique el contenido del archivo.

El comando anterior hace que todo lo que se recibe por la entrada estándar (teclado), pase a formar parte del contenido de un archivo llamado *lomo.* De esta forma se obtiene un editor muy sencillo, que se actualiza en disco una línea cada vez que se oprima la tecla *'enter'.*

Existe además otra forma de redireccionar la salida estándar y es que por medio de los caracteres '>>', antes de explicar la diferencia con respecto al carácter '>', escriba lo siguiente:

```
[usuario@geniux ~]$ cat lomo >> bashi.sh
```

Al consultar el contenido de *bashi.sh,* observará que el contenido del archivo *lomo,* se encuentra al final del archivo bashi.sh.

Ahora, realice la misma operación de la siguiente forma.

```
[usuario@geniux ~]$ cat lomo > bashi.sh
```

Al realizar la operación anterior, observará que el contenido de *lomo,* sobreescribe lo contenido en *bashi.sh.*

Como resultado de los ejercicios anteriores se puede concluir lo siguiente:

>	Sobreescribe el contenido del archivo destino.
>>	Agrega información al final del archivo destino.

Por último, en el siguiente ejemplo se muestra un truco clásico para borrar el contenido de un archivo sin borrarlo y volver a crearlo.

```
[usuario@geniux ~]$ cat /dev/null > bashi.sh
```

El archivo /dev/null es un archivo de dispositivo que se comporta como un archivo vacío, esto quiere decir que si se sobre escribe un archivo, con un archivo vacío, el resultado es que el archivo receptor (en esta caso bashi.sh) tambien quedará vacío.

Ahora, ha llegado el momento de poder separar la salida estándar del error estándar.

Hasta ahora no se ha considerado el hecho de que se puedan cometer errores a la hora de ejecutar las herramienta, pero en la siguiente ejecución, observaremos cómo separar el error estándar de la salida estándar.

```
[usuario@geniux ~]$ cat bashi.sh filemon.bat
```

Esta secuencia de comandos nos envía el contenido de *bashi.sh,* a pantalla y un error, que aclara la inexistencia de archivo *filemon.bat,* tal y como se muestra en el siguiente recuadro.

```
Nombre: Juan Peralta
Dirección: Calle de la Amargura #10
Edad: 25
Sexo: Masculino.
cat: filemon.bat: No existe el fichero o el directorio
```

Ahora desaparecerá el mensaje de error y únicamente quedará en pantalla el contenido de *bashi.sh,* al ejecutar lo siguiente:

```
[usuario@geniux ~]$ cat bashi.sh filemon.bat 2> error.txt
```

La ejecución anterior enviará el resultado libre de errores a la salida estándar y el error estándar se reenviará hacia el archivo *error.txt*.

Si observa el contenido del archivo error.txt y podrá comprobar lo explicado anteriormente.

Con esto queda plenamente demostrado que la salida estándar es un flujo de datos independiente al error estándar.

Para fines prácticos, cada vez que se desee redireccionar el error estándar, se escriben los caracteres "2>".

EJERCICIOS:

1. Determine el número de líneas que contiene el archivo */etc/ypserv.conf*
2. Ejecute la herramienta cat utilizando la opción -n, como se muestra en el siguiente recuadro y observe el resultado.

```
[usuario@geniux ~]$ cat -n .bashrc
```

3.1.13 LA HERRAMIENTA *more*

Al igual que la herramienta **cat,** esta herramienta permite mostrar la información de un archivo, la diferencia radica en el hecho de que este último al sobrepasar el límite de información que se puede desplegar en pantalla, hace una pausa hasta que se pulse ENTER o la barra espaciadora.

Formato: more <lista-de-archivos>

Opciones: -d Muestra al final de la pantalla la siguiente leyenda "[Presione la barra espaciadora para continuar; 'q' para salir.]"

El comando *more,* es muy similar a *cat,* con la diferencia que éste, muestra la información por pantallas.

Antes de continuar es importante aclarar que con esta herramienta también es posible hacer un redireccionamiento a la salida estándar y al error estándar así que vamos a probar lo siguiente:

Primero vamos a generar un archivo llamado *salida2.txt,* a partir de un redireccionamiento de un *ls -lRa,* al directorio */etc,* tal y como se muestra a continuación:

```
[usuario@geniux ~]$ ls - lRa /etc > salida2.txt
```

Ahora se aplicará el comando *more,* a este archivo de la siguiente forma:

```
[usuario@geniux ~]$ more salida2.txt
```

Para avanzar página por página, utilice la barra *espaciadora,* y para avanzar línea por línea, utilice la tecla *enter.*

3.1.14 LA HERRAMIENTA *less*

Esta herramienta permite mostrar un archivo por pantallas, pero a diferencia de la herramienta more permite desplazarse hacia arriba y hacia abajo del archivo mostrado.

Formato:	less <nombre-de-archivo>
Opciones:	No hay opciones.

La definición de *less,* en sí, no dice nada.La herramienta *Less,* permite ver el contenido de un archivo con la ventaja de poder desplazarse dentro de éste, tanto hacia el inicio, como hacia el final utilizando las flechas del teclado extendido o las teclas de desplazamiento por bloques.

Para comprobar su funcionamiento siga los siguientes pasos:

1. Verifique la existencia del archivo *telefono.txt.*
2. Aplique *less,* al archivo *telefono.txt,* tal como se muestra a continuación.

```
[usuario@geniux ~]$ less telefono.txt
```

Utilice las flechas de desplazamiento del cursor, así como las teclas de desplazamiento de bloque (PgUp ó PgDn), para explorar el archivo.

Para salir de la herramienta presione la tecla "q".

3.1.15 LA HERRAMIENTA *sort*

Esta herramienta toma el archivo indicado como argumento, y muestra en la salida estándar el resultado ordenado.

Formato:	sort [opciones | grupo de opciones] [lista-de-campos] [lista-de-archivos]	
Opciones:	-b	Ignora espacios iniciales en blanco
	-c	Solamente verifica el orden
	-f	No hace distinción entre mayúsculas y minúsculas.
	-n	Ordenamiento numérico.
	-r	Sentido inverso.
	-t	<carácter delimitador>
	-k n	Esta opción tiene que ir en conjunto con la opción -t. n es el número de fila a ordenar empezando desde 0.

Para poder comprender las prestaciones que nos ofrece esta herramienta, será necesario analizar los siguientes ejemplos:

Debe crearse el archivo *saat.txt,* cuyo contenido es el siguiente:

```
ACAMBAY
ACULCO
BOBINI
ACABARO
CEILAN
```

Para ordenar el contenido del archivo basta con ejecutar siguiente instrucción:

```
[usuario@geniux ~]$ sort -c saat.txt
```

El resultado es que ACAMBARO, está fuera de orden. En este caso, el argumento –c, sólo nos dice cuál de las palabras no está o se encuentra en orden.

Vamos a agregar ahora la palabra '*acapulco* ' al final del archivo, y aplicamos de nuevo la herramienta *sort,* como se muestra en el siguiente recuadro:

```
[usuario@geniux ~]$ sort saat.txt
```

El resultado es el envío a salida estándar de todas las palabras en orden, pero la palabra *acapulco,* no se encuentra ordenada de manera adecuada, esto se debe a que por defecto la herramienta *sort,* ordena tomando en cuenta el código ASCII. Para evitar este problema, hay que aplicar la opción "-f".

```
[usuario@geniux ~]$ sort -f saat.txt
```

Ahora es tiempo de probar la opción -r con el siguiente ejemplo:

```
[usuario@geniux ~]$ sort -rf saat.txt
```

La opción *–r,* nos devuelve el resultado en forma inversa; en este caso, primero aquellos que tienen el valor alfabético mayor, y al fin, los que tienen el valor alfabético menor.

Para la siguiente secuencia de ejemplos hay que crear un archivo llamado *papel.txt,* que debe contener lo siguiente:

```
10 Laura
45 Rosendo
88 Melquiades
 0 Saturnino
33 Pancho
 3 Ignacio
```

Al aplicar el comando *sort,* este archivo también nos devuelve el orden de los valores de forma equivocada, debido a que toma el valor ascii, de cada carácter, y no su valor numérico. La solución a este problema es utilizar la opción *–n,* a fin de que pueda tomar su valor numérico y nos devuelva el ordenamiento correcto.

Para comprobar esto ejecute la siguiente secuencia.

```
[usuario@geniux ~]$ sort -n papel.txt
```

Obviamente, si deseamos los valores de manera inversa, basta con aplicar la opción -r.

```
[usuario@geniux ~]$ sort -nr papel.txt
```

Ahora, hay que modificar un poco el archivo *papel.txt,* añadiendo el carácter ':', entre cada campo, como se muestra a continuación.

```
10:Laura
45:Rosendo
...
```

El resultado es que podemos ahora considerar al archivo de texto como una pequeña base de datos con 2 columnas y que utiliza como separador de columnas el caracter ":" (dos puntos).

Al aplicarle la herramienta con la opción -f:

```
[usuario@geniux ~]$ sort -f papel.txt
```

Se obtendrá el siguiente resultado:

```
0:Saturnino
10:Laura
33:Pancho
3:Ignacio
45:Rosendo
88:Melquiades
```

Pero este no es un resultado adecuado, ahora se tienen que utilizar las opciones -t y -k como se muestra en el recuadro siguiente.

```
[usuario@geniux ~]$ sort –t: -k 1 -n -f papel.txt
```

Como puede apreciarse, la utilización de la opción *–t,* nos indica que el archivo tiene como delimitador entre columnas el carácter dos puntos (*-t:*), *y* la opción -k 1, especifica que el ordenamiento se hará de acuerdo al primer campo de izquierda a derecha. Debido a que este campo es numérico, entonces utilizamos la opción -n.

Si deseamos ordenar el archivo por nombre, entonces se utilizan los siguientes parámetros:

```
[usuario@geniux ~]$ sort –t: -k 2 -f papel.txt
```

3.1.16 LA HERRAMIENTA *tail*

Despliega la última parte de un archivo. Toma la entrada del archivo especificado en la línea de mandato o de la entrada estándar.

Formato:	tail [opciones]... [nombre-de-archivo]...	
Opciones:	-f	(Actualiza el archivo aunque este no se cierre).
	-n	<num> (Mostrará la(s) ultima(s) linea(s) del archivo).

Vamos a comenzar por entender cómo funciona este comando, y para ello, es necesario generar un archivo llamado *archi.txt,* que contendrá una lista de los archivos y directorios, que se encuentran en el directorio raíz.

```
[usuario@geniux ~]$ ls - l / > archi.txt
```

Para observar su contenido, utilice el comando cat, y después aplique tail, de la siguiente forma:

```
[usuario@geniux ~]$ tail archi.txt
```

Al observar la salida se puede concluir que el comando *tail,* nos muestra las últimas diez entradas de un archivo. ¿Qué sucede si deseo ver únicamente las últimas dos entradas?

Para responder esta pregunta, ahora se utilizará un archivo llamado *"numeros.txt",* cuyo contenido es el siguiente:

```
uno
dos
tres
cuatro
cinco
seis
siete
ocho
nueve
diez
```

Aplíquese el comando tail, de la siguiente manera:

```
[usuario@geniux ~]$ tail -n2 numeros.txt
```

Como resultado se muestran las últimas dos líneas del archivo.

3.1.17 LA HERRAMIENTA *head*

Despliega la primera parte de un archivo. Toma la entrada del archivo especificado en la línea de comandos o de la entrada estándar.

Formato:	head [opciones]... [nombre-de-archivo]...	
Opciones:	-n <num>	(Muestra la(s) primera(s) línea(s) que se le indiquen).
	-c <num>	(Muestra solo el número de caracteres señalados).

Esta herramienta tiene el mismo comportamiento que *tail,* aunque la diferencia está en que *head* muestra la parte inicial de un archivo.

Para comprobarlo se puede ejecutar lo siguiente:

```
[usuario@geniux ~]$ head -n2 numeros.txt
```

Este ejemplo es similar al ejemplo anterior del *tail,* con la única diferencia que en lugar de mostrar la última parte, mostrará el principio del mismo.

Ahora, si se desea ver únicamente los primeros 10 caracteres del archivo se ejecuta esta herramienta de la siguiente forma:

```
[usuario@geniux ~]$ head - c10 numeros.txt
```

3.1.18 LA HERRAMIENTA *vi*

El vi, es un editor de texto eficaz, interactivo y orientado visualmente.

Formato:	vi [nombre-de-archivo]
Opciones:	+[num] al abrir el archivo el cursor se posicionará en la línea indicada por num.

El editor vi, es una de las herramientas más comunes en UNIX ®, y su uso es fácil. En un principio puede resultar incómodo, sobretodo, para aquellos usuarios acostumbrados a los ambientes de trabajo gráficos.

Para iniciarse en el uso de esta herramienta, crear un archivo llamado *estados.txt,* en él, se deben escribir del lado izquierdo, todos los estados del país y a lado derecho, el nombre de su respectivas capitales.

A continuación se muestra un ejemplo del formato que debe tener este archivo:

```
Aguascalientes          Aguascalientes
```

Para abrir un archivo nuevo desde vi, hay que escribir lo siguiente:

```
[usuario@geniux ~]$ vi estados.txt
```

Al iniciar el editor verá en pantalla algo como lo que se ilustra a continuación:

```
~
~
~
"estados.txt" [New File]
```

Para poder iniciar la escritura hay que oprimir la letra *'i'* o la tecla *'Insert'*. Al hacerlo, observará en la parte inferior izquierda lo siguiente:

```
~
~
~
-- INSERTAR --
```

Ahora ya es posible comenzar a editar el archivo.

La herramienta vi, funciona mediante modos de operación, cuando aparece la palabra --INSERTAR--, se entra en modo de inserción. Para salir de este modo se debe oprimir la tecla "*ESC*".

Para la edición de un archivo se dispone de tabuladores, espacios en blanco, mayúsculas y minúsculas, etc. Como se mencionó anteriormente, *vi,* es un editor muy sencillo de utilizar.

Cuando se desea guardar el contenido del archivo, se sale del modo de inserción (tecla *ESC*), y se escribe lo siguiente:

```
~
:w
```

Asegúrese que la 'w', sea minúscula, y que los ':' aparezcan en la esquina inferior izquierda.

Otra función básica que debe conocer, es cómo salir del editor, y para ello, basta con escribir lo siguiente:

```
~
~
~
:q
```

Asegúrese de haber guardado el contenido del archivo; en caso contrario, no se le permitirá salir del editor. Si desea salir sin guardar los cambios, hay que escribir lo siguiente:

```
~
~
~
:q!
```

A continuación se ilustra una tabla con algunos de los mandatos válidos y mas comunes para el editor vi.

i	Activa el modo de inserción. En algunos casos la tecla de *insert,* funciona de la misma manera al pulsarla una sola vez.
:w	Permite guardar los cambios.
:q	Salir del *vi,* sin haber hecho cambios.
:wq	Guarda y sale, del *vi.*
:q!	Salir de *vi,* sin guardar cambios.
:x	Guarda y sale, del *vi.*
:u	Deshacer el último cambio al documento.

Entre las características de un editor texto, debe tenerse la facilidad de localizar una palabra o reemplazarla, copiar, pegar y eliminar lineas. Para eso *vi,* tiene los siguientes mandatos:

/<patrón a buscar palabra>	Permite buscar hacia adelante del archivo la palabra o patrón específico.
? <patrón a buscar palabra>	Permite buscar hacia atrás del archivo una palabra o patrón específico.

Para poder trabajar con las características de copiar, pegar y eliminar lineas, tendrá que salir del modo de inserción [tecla Esc] y realizar las siguientes secuencias del teclado.

yy ó nyy	Permite copiar una linea en la posición del cursor. Si desea copiar *n* lineas, deberá teclear primero el número de lineas que se desea mas dos veces *y.*
p	Permite pegar *n* lineas según lo almacenado con la opción *yy* a partir de la posición del cursor.
dd ó ndd	Permite eliminar una linea en la posición del cursor. Si desea eliminar *n* lineas, deberá teclear primero el número de lineas mas dos veces *d.*

3.1.19 LA HERRAMIENTA *ln*

Crea un enlace a un archivo ordinario y tiene dos modos de operación: El primero, hace un hacia un archivo ya existente, el segundo, hace un enlace hacia un directorio.

Formato:	ln [opciones] archivo-existente [nueva liga] ln [opciones] directorio-existente directorio
Opciones:	-s liga suave

Una liga es el equivalente a un acceso directo, pero con la diferencia que aquí se realiza en texto.

¿Recuerda el archivo *estados.txt*? Antes de continuar es importante hacer una copia de respaldo tal y como se muestra en el siguiente recuadro.

```
[usuario@geniux ~]$ cat estados.txt > estados.antes.txt
```

Para poder continuar, reconstruya el árbol de directorios de la figura 8, y mueva el archivo *estados.txt,* hacia el directorio *user2*. Después se debe crear una liga llamada *states.txt,* que apunte hacia *estados.txt,* de la siguiente manera:

```
[usuario@geniux ~]$ ln home/user2/estados.txt states.txt
```

Con el ejemplo del recuadro anterior se esta haciendo una liga llamada *states.txt* que apunta hacia el archivo home/user2/estados.txt.

Ahora, para ver los atributos, Haga un listado sobre los archivos *states.txt,* y *home/user2/estados.txt.*

La liga creada en el ejemplo anterior se les conocen como ligas fuertes, las ligas fuertes permiten tener acceso a ciertos archivos desde distintos directorios sin duplicar información. Es como tener duplicado el archivo sin ocupar espacio adicional en disco.

Además las ligas fuertes tienen algunas propiedades como por ejemplo, si se elimina cualquiera de los accesos al archivo, la información no se pierde debido a que tiene acceso a este por cualquiera de los accesos restantes.

Para comprobarlo realice el siguiente ejercicio:

1.- Borre el archivo *estados.txt,* del directorio *user2.*

```
[usuario@geniux ~]$ rm  home/user2/estados.txt
```

Si usted consulta a *states.txt,* podrá seguir viendo su contenido, debido a que es una liga dura.

2.- Ahora, elimine el archivo *states.txt.*

```
[usuario@geniux ~]$ rm states.txt
```

Ahora, debido a que se ha borrado la única liga que apuntaba hacia la información, ya no se tiene acceso al contenido del archivo y este se pierde.

Repita toda la operación anterior desde copiar el archivo *estados.txt,* hasta borrar *estados.txt,* pero esta vez cree una liga suave de la siguiente manera.

```
[usuario@geniux ~]$ ln -s home/user2/estados.txt  states.txt
```

Con la opción -s es posible crear una liga suave. Estas ligas tienen como característica que, cuando se consulta la liga del archivo en su forma larga (ls -la), muestra la siguiente información.

```
lrwxrwxrwx 1 root root  7 Apr 26  2008 S10sshd -> ../sshd
[usuario@geniux ~]$
```

El caracter "**l**" al inicio de la máscara de derechos nos dice que es una liga y los caracteres "**->**" antes del nombre del archivo nos indica que trata de una liga suave.

EJERCICIOS:

Se deja como ejercicio responder a las siguientes preguntas:

1.- ¿Se pueden crear ligas duras sobre directorios?
2.- ¿Se pueden crear ligas suaves sobre directorios?

3.1.20 LA HERRAMIENTA *chmod*

Cambia las formas en que el propietario, el grupo y/o los demás usuarios pueden acceder a un archivo o directorio. Sólo el propietario de un archivo o el súper usuario puede cambiar el modo de acceso a un archivo.

Formato:	chmod quién [operación] [permiso] lista-de-archivo. chmod mascara lista-de-archivos.
Opciones:	-c (cambio)
	-R (recursivo)

Con esta herramienta se pueden modificar los derechos que aparecen en la máscara de derechos y el efecto resultante es una modificación en los permisos para tener acceso a un archivo o directorio.

Los posibles argumentos son los siguientes:

Quién:

u	(usuario) dueño del archivo.,
g	(grupo) grupo al que pertenece el archivo.
o	(otros) los demás usuarios.
a	(todos) puede usarse en lugar de u, g y o.

Operación:

+	agrega el permiso para el usuario especificado.
-	quita el permiso para el usuario especificado.
=	fija el permiso para el usuario especificado; todos los demás permisos para este usuario se habilitan de nuevo.

Permiso:

r	permiso de lectura.
w	permiso de escritura.
x	permiso de ejecución.

Otra forma de asignar derechos, más sencilla es modificando la máscara de derechos de manera numérica utilizando los siguientes valores.

1	permiso de ejecución.
2	permiso de escritura.
4	permiso de lectura.

También son válidas sus combinaciones:

6	permisos de lectura y escritura
7	permisos de lectura, escritura y ejecución.

Estos permisos pueden ser aplicados a archivos, directorios o dispositivos y numéricamente se aplican a los 3 bloques de la máscara de derechos: **u**suario,**g**rupo,**o**tros.

Para comprender mejor esta herramienta, a continuación se muestran los siguientes ejemplos:

1.-Crear el archivo *prog.sh* y eliminar todos sus permisos.

```
[usuario@geniux ~]$ chmod a-wrx prog.sh
```

De forma numérica, el proceso es el siguiente.

```
[usuario@geniux ~]$ chmod 000 prog.sh
```

Liste el archivo y observe el resultado. Puede preguntarse tal vez, cual es el efecto sobre el archivo al tener estos derechos, para comprobarlo, intente copiar la información de un archivo hacia *prog.sh*.

```
[usuario@geniux ~]$ cat numeros.txt > prog.sh
```

Esto produce un mensaje de error debido a que el archivo no tiene derechos de escritura. Para corregir esto hay que realizar el siguiente ejercicio.

2.- Asignar permisos de escritura al propietario del archivo *prog.sh*.

```
[usuario@geniux ~]$ chmod u+w prog.sh
```

De manera numérica:

```
[usuario@geniux ~]$ chmod 200 prog.sh
```

Al tratar de copiar la información,no debe existir problema, pero al intentar ver el contenido de *prog.sh,* utilizando la herramienta cat, observará que no se puede y esto

se debe a que no tiene derechos de lectura, por lo tanto, hay que dar derechos de lectura al archivo. Para ello realice el ejercicio No 2.

3.-Asignar al propietario del archivo *prog.sh*, permisos de lectura.

```
[usuario@geniux ~]$ chmod u+r prog.sh
```

En formato numérico, se aplica de la siguiente manera.

```
[user@geniux user]$ chmod 400 prog.sh
```

Ahora sí es posible ver su contenido.

Para concluir con el estudi básico de esta herramienta, hay que hacer un pequeño programa.

Para esto, utilizando *vi,* borre el contenido de *prog.sh,* y escriba lo siguiente:

```
echo "Programa No 1"
echo "- - - - - - - - - - - - - - - - - - - -"
ls - l
echo "- - - - - - - - - - - - - - - - - - - -"
```

Para ejecutar el programa, se invoca de la siguiente manera:

```
[usuario@geniux ~]$ ./prog.sh
```

Como puede suponer, no funcionó debido a que no tiene derechos de ejecución para el propietario del archivo, así que la solución es darle los permisos adecuados de la forma siguiente:

```
[usuario@geniux ~]$ chmod u+x prog.sh
```

Utilizando el formato numérico.

```
[usuario@geniux ~]$ chmod 100 prog.sh
```

Ejecútelo y verá que ahora ¡sí se puede!

Esta es la razón por la que no existen los archivos .exe y/o .com, tal y como sucede en MS. Windows ®, aquí, para poder hacer un archivo "ejecutable", basta con asignarle sus respectivos derechos de ejecución.

3.1.21 LA HERRAMIENTA *chown*

Fija un nuevo propietario para un archivo.

Formato:	chown <nuevo-propietario> archivo.
Opciones:	-R Recursivo.

chown, permite cambiar el propietario a un archivo. Esta operación solamente puede ser realizada por el usuario *root*.

Recuerde que, para comprobar el comportamiento de este comando, hay que ingresar al sistema como usuario *root* y realizar los siguientes ejercicios.

1.-Haga una lista de forma larga en el directorio actual y observe que el propietario de cada archivo.

Utilizando la herramienta touch, hay que crear un archivo llamado *prog1.sh*. Este archivo debe tener como propietario al usuario *root*.

Ahora, asignar como propietario al usuario nobody de la siguiente manera.

```
[root@geniux ~]# chown nobody prog1.sh
```

Para comprobarlo liste el archivo de forma larga.

También es posible asignar el propietario a un directorio y todos los archivos que lo contienen. Para entender mejor esto , realice el siguiente ejercicio.

2.- Siendo el usuario root, crear un directorio llamado canelo y a dentro de este crear los siguientes archivos: chinelo.txt, anhelo.bat y franelo.bak y cambiar su propietario al usuario *nobody* con una sola instrucción.

Ya creado este directorio y los archivos al listarlo en su forma larga se debe observar lo siguiente:

```
drwxr-xr-x   2    root root 4096    May 23 19:48 .
drwx------  46    root root 4096    May 23 19:43 ..
-rw-r--r--   1    root root    0    May 23 19:43 anhelo.txt
-rw-r--r--   1    root root    0    May 23 19:43 chinelo.txt
-rw-r--r--   1    root root    0    May 23 19:43 franelo.bak
```

Para modificar el propietario del directorio canelo y sus archivos al usuario *nobody*, se debe ejecutar la siguiente instrucción.

```
[root@geniux ~]#chown -R nobody canelo
```

La opción -R modifica de manera recursiva el propietario a un directorio así como los archivos y directorios contenidos en este.

3.1.22 LA HERRAMIENTA *chgrp*

Fija un nuevo grupo para un archivo.

Formato: chgrp <archivo(s)>

Opciones: -R (recursivo)

Su uso es exactamente igual que *chown,* pero con la diferencia que modifica el grupo propietario del archivo.

Para comprender mejor su uso, realice el siguiente ejemplo:

1.- Siendo el usuario *root,* modifique el archivo *prog.sh,* para que el grupo propietario sea el grupo *nobody.*

```
[root@geniux ~]# chgrp nobody prog.sh
```

2.- Asigne al directorio canelo y sus archivos el grupo nobody.

```
[root@geniux ~]#chgrp -R nobody canelo
```

3.1.23 LA HERRAMIENTA *find*

Busca y selecciona los archivos que están en los directorios especificados y que se describen con una expresión.

Formato:	find [directorio en donde buscar] [opciones | [grupo de opciones] expresión.	
Opciones:	-name	Nombre del archivo a buscar o expresión regular.
	-iname	Nombre del archivo a buscar o una expresión regular, la expresión será evaluada tanto para mayúsculas como para minúsculas.
	-type	Tipo de archivo a buscar.
	b	Archivo especial de bloque.
	c	Archivo especial de caracter.
	d	Directorio.
	f	Archivo ordinario.
	l	Liga.
	p	Tubería con nombre.
	-user	Archivos que pertenecen al usuario especificado.
	-group	Archivos que pertenecen al grupo especificado.
	-links	Archivos que poseen el número de enlaces especificados.
	-size	Archivos que coinciden con el tamaño en bytes.

La herramienta *find,* es sin duda un elemento que no puede faltar en un sistema de archivos cuyo árbol de directorios sea complejo.

Para aclarar cómo puede aprovecharse se plantearán los siguientes ejemplos:

1.- Localizar el archivo *resolv.conf,* desde el directorio raíz.

```
[usuario@geniux ~]$ find /  -name resolv.conf
```

Se puede apreciar que la sintaxis es bastante clara: "*se buscará desde el directorio raíz, y en todos los subdirectorios, aquellos archivos cuyo nombre sea resolv.conf*". También es válido el uso de comodines, como por ejemplo resolv*.

2.- "Localizar el archivo *resolv.conf,* desde el directorio */etc* ".

```
[usuario@geniux ~]$ find /etc/  -name resolv.conf
```

3.- "Localizar los dispositivos de bloque en el directorio */dev* ".

```
[usuario@geniux ~]$ find /dev  -type b
```

4.- "Localizar los archivos cuyo propietario tenga el uid 309, en el directorio actual".

```
[usuario@geniux ~]$ find -user 309
```

EJERCICIOS:

1.- Buscar en todo el disco duro aquellos archivos o directorios cuyo nombre empiecen con la palabra *Apach.*

2.- Dentro de su directorio personal de usuario buscar al archivo *estados.txt.*

3.- Dentro del directorio actual buscar todos los subdirectorios.

4.- Dentro del directorio actual buscar todas las ligas.

5.- Desde el directorio raíz buscar todos los archivos en el disco duro que pertenezcan al usuario que está utilizando en este momento.

Al hacer búsquedas en todo el disco se generan una gran cantidad de errores. Elimínelos enviando la salida estándar hacia el archivo /dev/null.

El archivo */dev/null* es un archivo de dispositivo con caracteríasticas muy especiales. No está asociado con algún dispositivo físico, y al recibir texto lo envía hacia "ningún lado", es por ello que equivale a lo que coloquialmente conocemos como un "barril sin fondo".

3.2 HERRAMIENTAS PARA LA MANIPULACIÓN DE TEXTO.

En las secciones anteriores se han analizado herramientas para la manipulación de archivos de texto. Herramientas tales como vi, less, cat, more , etc.

Sin embargo ahora se presentan una serie de herramientas que nos permitirán entre otras cosas, extraer información ya sea un archivo, un conjunto de archivos o de la salida de un comando.

3.2.1 LA HERRAMIENTA *grep*

Busca un patrón en uno o más archivos, línea por línea. El patrón puede ser una cadena de caracteres sencilla o cualquier otra forma de expresión regular.

Formato:	grep [opciones | grupo de opciones] patrón [lista de archivos]	
Opciones:	-v	Invierte el significado de la prueba.
	-n	Imprime el número de línea en el archivo.
	-c	Devuelve el número de líneas que posee el patrón.
	-w	Busca una palabra en especifico y no un patrón.
	-i	Ignora entre mayúsculas y minúsculas.

Con esta herramienta es posible localizar información mas específica dentro de un archivo de texto. La búsqueda se realiza por medio de expresiones regulares.

Una expresión regular consiste en una expresión que describe una cadena o un conjunto de cadenas .

A continuación se muestran los símbolos con los cuáles podemos construir *expresiones regulares* usando la herramienta *grep*.

Símbolo	Significado	Ejemplo	Identificados
.	Identifica cualquier carácter	chil.	*chili, chile*
*	Identifica cero o más repeticiones del carácter precedente.	ap*le	*aple, apple*
[]	Identifica a cualquiera de los caracteres incluidos en los corchetes.	[Cc]hicken	*Chicken, chicken*

[a-z]	Identifica cualquier carácter en el rango especificado.	[A-Z, a-z]	*Cualquier cadena alfabética.*
^	Comienzo de línea	^beef	*beff a comienzo de una línea.*
$	Fin de línea	soup$	*soup al final de una línea.*

La herramienta *grep* será utilizado de aquí en adelante para búsqueda de textos o palabras dentro de archivos.

Como introducción,realice el siguiente ejemplo:

1.- crear un archivo llamado *grapa.txt,* cuyo contenido se muestra en el siguiente recuadro.

```
pollo
gato
perro
ardilla
armadillo
membrillo
perdiz
hombre
hombro
pollino
```

Para entender el funcionamiento de *grep,* vamos a generar expresiones de búsqueda:

a).- Mostrar todas aquellas palabras que contengan la letra "o" dentro del archivo *grapa.txt.*

```
[usuario@geniux ~]$ grep o grapa.txt
```

En este ejemplo, existe un pequeño truco debido a que *grep,* selecciona en realidad aquellas líneas de texto que corresponden al patrón buscado, y debido a que las palabras están una por cada línea entonces selecciona todas aquellas palabras que cumplen con el patrón especificado.

b).- Mostrar todas aquellas palabras que empiecen con la letra "p".

```
[usuario@geniux ~]$ grep ^p grapa.txt
```

Aquí, el carácter "^", como puede ver en la tabla anterior, significa todo lo que empieza con el carácter o el texto indicado.

c).- Obtener todas aquellas palabras que terminen con "br", seguido de un carácter alfabético.

```
[usuario@geniux ~]$ grep br[a-z]$ grapa.txt
```

d).-Crear una expresión regular que seleccione únicamente las palabras ardilla y membrillo.

```
[usuario@geniux ~]$ grep ^[am][er][dm][ib].*ll[ao]$ grapa.txt
```

Ahora realice los siguientes ejercicios:

EJERCICIOS:

1.- Crear una expresión regular que seleccione únicamente las palabras *armadillo* y *ardilla* dentro del archivo *grapa.txt*.

2.- Utilizando un archivo auxiliar, seleccione los nombres de todos aquellos subdirectorios que se encuentren dentro de su directorio de trabajo.

3.- Utilizando un archivo auxiliar, seleccione y guarde de su directorio personal, todos aquellos que sean ligas.

4.- Mostrar todos los elementos del archivo *grapa.txt*, excepto las palabras *pollo* y *pollino*.

5.- Eliminar del archivo *grapa.txt*, las palabras *pollo* y *pollino*.

6.- Seleccionar del archivo *telefono.txt*, todos aquellos cuyo apellido sea *GONZALEZ*.

7.- Buscar dentro del archivo *telefono.txt*, aquellos cuyo apellido sea *GONZALEZ* y vivan en la *Colonia progreso*.

8.- Buscar en el archivo *telefono.txt,* aquellas personas cuyos apellidos sean *Martinez Perez.*

9.- Localizar en el archivo *telefono.txt,* todos los códigos postales cuyo dígito, en la parte de las decenas, sea impar.

10.- Seleccionar y Guardar en un archivo auxiliar en la parte inicial todos los códigos postales que no sean "0" y en la segunda parte todos los demás.

3.2.2 HERRAMIENTA *cut*

Imprime secciones de cada línea de entrada.

Formato:	cut [opciones | grupo de opciones] nombre-de-archivos	
Opciones:	-b	Lista de bytes a ser cortados
	-f	No de campo a ser cortado
	-d	Delimitador de campo.

Esta herramienta permite cortar porciones de un archivo de texto de manera vertical .

La mejor manera de comprender esta herramienta es probando los siguientes casos.

1.- Obtener el segundo caracter de cada línea en el archivo *telefono.txt.*

```
[usuario@geniux ~]$ cut -b 2 telefono.txt
```

Esta herramienta ejecuta la acción especificada sobre cada una de las líneas del archivo especificado:

2.- Para el ejemplo siguiente es necesario crear los archivos *s.db, p.db, sp.db.* Su contenido se ilustra en la figura 9.

s.db	*p.db*	*sp.db*
S1:SALAZAR:20:ACAPULCO	P1:TUERCA:ROJO:12:ACAPULCO	S1:P1:300
S2:JAIMES:10:TAXCO	P2:PERNO:VERDE:17:TAXCO	S1:P2:200
S3:BERNAL:30:TAXCO	P3:BIRLO:AZUL:12:ZIHUATANEJO	S1:P3:400
S4:CORONA:60:ACAPULCO	P4:BIRLO:ROJO:14:ACAPULCO	S1:P4:200
S5:ALDAMA:50:IGUALA	P5:LEVA:AZUL:12:TAXCO	S1:P5:100
S6:ESPIRITU:80:CHILPO	P6:ENGRANE:ROJO:19:ACAPULCO	S1:P6:500
		S2:P1:300
		S2:P2:800
		S3:P2:200
		S4:P2:200
		S4:P4:300
		S4:P5:400

Figura 9. Contenidos de archivos s.db, p.db, sp.db

Las tablas s.db, p.db y sp.db, se pueden considerar como las tablas de proveedores, partes y envíos de una base de datos. A continuación se presentan algunas búsquedas que se realizarán en estas tablas, aprovechando la herramienta cut.

A) Obtener los identificadores de los proveedores en tabla *s.db*.

```
[usuario@geniux ~]$ cut -d: -f1 s.db
```

Para la herramienta cut, es importante indicarle cuál será el separador de columna para poder establecer la diferencia entre ellas. En este ejemplo, el separador, son los dos puntos ":", y se especifica con el parámetro -d.

B) Obtener de la tabla *p.db*, el nombre de cada parte, su color y la ciudad de su ubicación.

```
[usuario@geniux ~]$ cut -d: -f2,3,5 p.db
```

Con esta expresión se corta del archivo *p.db,* sólo las columnas 2,3 y 5 delimitados por los dos puntos.

C) Obtener del archivo *p.db*, toda su información excepto la ciudad donde se ubican las partes.

```
[usuario@geniux ~]$ cut -d: -f 1- 4 p.db
```

De esta forma se expresa un rango de columnas a cortar.

Hay que recordar siempre que, las columnas se empiezan a contar a la izquierda del primer delimitador y se cuentan de izquierda a derecha, a partir del número uno.

3.2.3 LA HERRAMIENTA paste

Imprime las líneas que consisten en una secuencia correspondiente de líneas de distintos archivos.

Formato:	paste [opciones | grupo de opciones] lista-de-archivos	
Opciones:	-d <delimitador>	Pega las líneas utilizando el delimitador.
	-s	Pega las líneas uniéndolas en una sola.

Este comando pega las líneas de un archivo con otro. Para comprender mejor como funciona hay que realzar los siguientes ejercicios.

1.- Utilizando la herramienta paste, unir verticalmente los archivos *s.db* y *p.db*.

```
[usuario@geniux ~]$ paste s.db p.db
```

La operación que aparece en el recuadro anterior, une ambos archivos línea por línea. Esto podría parecer obvio debido a que ambos archivos tienen el mismo número de líneas. Pero, ¿qué sucederá si el número de líneas no coincide?

Elimine una de las líneas del archivo *s.db*, y verifíquelo.

2.-Unir los archivos *s.db y p.db,* de manera que las columnas resultantes, sean separadas por el carácter "@" (arroba).

```
[usuario@geniux ~]$ paste -d"@" s.db p.db
```

3.2.4 LA HERRAMIENTA *join*

Esta herramienta permite unir archivos tal y como lo hace la herramienta *paste*, pero con la diferencia que las líneas se unen con base en la columna que tengan en común.

Formato:	join [opciones | grupo de opción] lista-de-archivos	
Opciones:	- j <num>	Número de campo a unir.
	-t	Carácter separador de entrada y salida.

Para poder comprender la lógica bajo la cuál opera esta herramienta analicemos el siguiente caso:

Como puede observar las tablas *s.db* y *sp.db*, comparten una misma columna, por lo tanto si queremos unir la tabla *sp.db* con la tabla *s.db*, por cada linea del archivo *sp.db* en donde coincida la columna se unirá una del archivo *s.db*. Para comprobarlo ejecutaremos *join* de la siguiente forma:

```
[usuario@geniux ~]$ join -t':' -1 1 -2 1 sp.db s.db
```

```
S1:P1:300:SALAZAR:20:ACAPULCO
S1:P2:200:SALAZAR:20:ACAPULCO
S1:P3:400:SALAZAR:20:ACAPULCO
S1:P4:200:SALAZAR:20:ACAPULCO
S1:P5:100:SALAZAR:20:ACAPULCO
S1:P6:500:SALAZAR:20:ACAPULCO
S2:P1:300:JAIMES:10:TAXCO
S2:P2:800:JAIMES:10:TAXCO
S3:P2:200:BERNAL:30:TAXCO
S4:P2:200:CORONA:60:ACAPULCO
S4:P4:300:CORONA:60:ACAPULCO
S4:P5:400:CORONA:60:ACAPULCO
```

El resultado devuelve todas las combinaciones de la tabla *sp.db* y *s.db*, que tengan en común el valor de la columna 's' (en ambos archivos la columna 1).

Si se analiza el resultado, por cada repetición de S1 en sp.db, aparece una copia de la línea S1 en s.db debido a que tienen en común la columna 1.

3.2.5 LA HERRAMIENTA tr

Permite cambiar caracteres de un archivo o eliminarlos al recibirlos desde la entrada estándar.

Formato:	tr [opciones | grupo de opciones] < archivo	
Opciones:	-d <carácter>	Indica el número de campo a unir.
	-s <car1><car2>	Permite sustituir cualquier cantidad de caracteres indicados por car1, por un solo caracter de car2.

Esta herramienta permite cambiar caracteres de un archivo por otros indicados.

Para comprobarlo, basta con probar los siguientes casos:

1.- Utilizando el archivo *s.db,* cambiar todos los caracteres de mayúscula a minúscula, utilizando la herramienta *tr,* tal y como se muestra a continuación:

```
[usuario@geniux ~]$ tr A-Z a-z < sp.db
```

2.- Utilizando el mismo archivo, eliminar todas las letras que se encuentre en el.

```
[usuario@geniux ~]$ tr -d A-Z < sp.db
```

3.- Substituir todos los números por el carácter ":"

```
[usuario@geniux ~]$ tr -s 0-9 ":" < sp.db
```

Por definición *tr,* recibe la información desde la entrada estándar, así que para ello, es necesario redireccionar el contenido de un archivo con el carácter '<'.

3.2.6 LA HERRAMIENTA *uniq*

Permite encontrar y omitir líneas de texto repetidas.

Formato:	*uniq* archivo.
Opciones:	Sin opciones.

Esta herramienta permite encontrar líneas que se repitan en un archivo de texto y mostrará el resultado con una sola coincidencia. Los siguientes ejemplos ilustran mejor el comportamiento de esta herramienta:

1.- Suponga que utilizando el archivo *sp.db*, se desea determinar cuáles son los proveedores que han hecho algún tipo de operación hasta la fecha.

Para obtener esta información, primero hay que ordenar el archivo y el resultado se guardará en un archivo auxiliar llamado *aux1*.

```
[usuario@geniux ~]$ sort sp.db > aux1
```

Como segunda etapa se cortará sólo la parte del identificador del proveedor y el resultado se enviará a un segundo archivo auxiliar llamado *aux2*.

```
[usuario@geniux ~]$ cut -d: -f1 aux1 > aux2
```

Por último, se aplicará la herramienta *uniq*.

```
[usuario@geniux ~]$ uniq aux2
```

El resultado final debe ser el siguiente:

```
S1
S2
S3
S4
```

De este ejemplo puede concluirse que el comando *uniq*, elimina palabras repetidas siempre y cuando estas, aparezcan consecutivos y en líneas de texto diferentes. Por ello primero se aplicó la herramienta *sort*.

EJERCICIOS:

1.-Obtener una lista que contenga el carácter inicial de cada una de las líneas del archivo *telefono.txt*, sin repeticiones.

3.2.7 HERRAMIENTA *wc*

Permite contar palabras, líneas y caracteres, en el archivo, enviando el resultado a la salida estándar.

Formato:	wc [opción | grupo de opciones] archivo
Opciones:	-c Número total de caracteres
	-l Número total de líneas.
	-w Número total de palabras.

Como se menciona en el resumen, esta pequeña herramienta cuenta el número de palabras, líneas y caracteres. Pudiera parecer una simple curiosidad, aunque probablemente se aprecien mejor sus cualidades al estudiar los siguientes casos:

1. -¿Qué devuelve la siguiente instrucción?

```
[usuario@geniux ~]$ wc telefono.txt
```

La salida envía a pantalla la siguiente información:

```
22419 204325 2802484 telefono.txt
```

Estos números, son el número de líneas, palabras y caracteres (respectivamente), que contienen el archivo *telefono.txt.*

2.- Imagine que ahora se desea saber cuántos registros tiene el archivo *telefono.txt*. Contarlos manualmente sería muy tardado (y aburrido), al hacerlo por medio de *cat,* hay que esperar que muestre todo el contenido del archivo (por lo tanto no es la opción más adecuada). ¿Por qué no intentar lo siguiente?

```
[usuario@geniux ~]$ wc -l telefono.txt
```

Nos devuelve el número de líneas y por ende de registros que posee el archivo *telefono.txt.*

3.3 AGRUPACION DE ÓRDENES.

Permite que dos o más órdenes sean escritas en la misma línea de comandos y ejecutadas de manera separada. Esta es una característica muy sencilla del intérprete de comandos que se está utilizando (bash), que es importante conocer.

Entrando ya en el tema de la agrupación de órdenes, imagínese ahora que desea saber las características del archivo *telefono.txt,* a través de un *ls, y* al mismo tiempo conocer el número de líneas que tiene.

Una forma sencilla de hacerlo es ejecutando por separado las herramientas ls y wc tal y como se muestra a continuación.

```
[usuario@geniux ~]$ ls -lh telefono.txt
```

y

```
[usuario@geniux ~]$ wc -l telefono.txt
```

Otra forma my sencilla de lograrlo, es agrupando las órdenes de la siguiente manera:

```
[usuario@geniux ~]$ ls -lh telefono.txt ; wc -l telefono.txt
```

La única diferencia está en el hecho de poder escribir ambas herramientas con sus respectivos parámetros, en una misma línea y ambas, se ejecutarán de izquierda a derecha.

Otro ejemplo podría ser, el buscar en el directorio de trabajo todos los archivos mayores de 1024k, y generar una lista de ellos, ordenados de mayor a menor por tamaño, y por el total de los archivos encontrados.

La secuencia es la siguiente:

```
[usuario@geniux ~]$ find . -size +1024k -printf "%s\t%p\n" > archivo ; sort
archivo > ultimo; wc -l archivo >> ultimo;
```

Es como hacer un pequeñísimo programa aprovechando las capacidades de las herramientas involucradas. Si se visualiza el contenido del archivo *ultimo,* se obtendría algo como esto:

```
1142321 ./prueba.txt.tar.com
1142321 ./test/prueba.txt.tar.com
1142321 ./todo.tar.com
1142336 ./prueba.txt.tar.gz
...
11 archivo
```

Para concluir, imagine que desea obtener el mismo resultado, pero ahora utilizando el menor número de elementos, además de borrar el archivo auxiliar *(archivo)*.

```
[user@geniux user]$ { find . -size +1024k -printf "%s\t%p\n" > archivo; sort
archivo; wc -l ultimo; rm -f archivo ; } > ultimo
```

Para comprobar el resultado, puede utilizarse el mismo procedimiento del ejercicio anterior.

Como puede observarse, la diferencia la marca el uso de los caracteres '{' y '}'.

3.4 EVALUACIÓN DE ÓRDENES

Bash también permite realizar instrucciones condicionales para ejecutar las herramientas:

Por ejemplo:

¿Qué resultado se obtiene al ejecutar la siguiente secuencia?

```
[usuario@geniux ~]$ ls -l a122* && rm *
```

Lo que se obtiene es una secuencia condicional de eventos, esto quiere decir que si se ejecuta satisfactoriamente el evento de la izquierda, entonces se ejecuta el de la derecha.

Ahora analizaremos la siguiente secuencia:

```
[usuario@geniux ~]$ ls a123* || touch a1234
```

Este caso es muy similar al anterior, pero ahora al ejecutar la secuencia de herramientas, la primera de la izquierda envía un mensaje de error, pero a pesar de esto, la segunda ejecuta lo que tiene que hacer.

En resumen, se puede decir lo siguiente:

Los caracteres &&, son el equivalente a un AND, debido a que la secuencia a su derecha se ejecuta siempre y cuando la secuencia de la izquierda se realice satisfactoriamente.

Los caracteres | |, actúan como un OR, debido a que sin importar si la secuencia de la izquierda se ejecutó con éxito o no, siempre se ejecutará la secuencia de la derecha.

EJERCICIOS:

1.- Determinar que hace la siguiente secuencia.

```
[usuario@geniux ~]$ ls a246* || touch a2468 && ls -lh a2468
```

3.5 MANEJO DE TUBERÍAS

Hasta ahora se ha utilizado una gran cantidad de comandos aislados y lo único que se ha logrado modificar es la salida estándar y el error estándar. Pues bien, ahora en esta sección aprenderá el cómo es posible encadenar las distintas herramientas y de esta manera empezar a contruir herramientas mas complejas.

A este concepto se le conoce como manejo de tuberías o si quiere verlo así: "entubamiento de herramientas".

En los recuadros siguientes se ilustran algunos ejemplos de cómo encadenarlas.

1.- A partir del directorio actual, obtener un listado de archivos del directorio actual y todos sus subdirectorios, además debo poder desplazarme entre ellos hacia arriba y hacia abajo. Todo esto con una sola secuencia de comandos.

```
[usuario@geniux ~]$ ls -lR | less
```

Aquí el carácter " | " (pipe), permite que la salida del comando *ls,* sea enviada a la herramienta *less,* para su procesamiento. De esta manera obtenemos un comando que me permite revisar manualmente que archivos tenemos en el directorio actual y en sus subdirectorios.

2.- Obtener el número total de directorios y subdirectorios, en el directorio actual.

```
[usuario@geniux ~]$ ls -lR | grep ^d | wc -l
```

Analizando la secuencia de herramientas en el recuadro anterior podemos observar que primero obtenemos todos los archivos y directorios dentro del directorio actual, el resultado se envía a la herramienta *grep,* donde se "filtran" y obtienen únicamente aquellos que son directorios, para finalmente contarlos con la herramienta *wc.* El resultado final es un valor numérico entero, que no dice cuantos directorios hay en el directorio actual.

3.- Obtener el número de personas en el archivo *telefono.txt,* cuyos apellidos sean primero DOMINGUEZ, y después PEREZ.

```
[usuario@geniux ~]$ grep DOMINGUEZ.*PEREZ telefono.txt|wc -l
```

4.- Obtener el número total de directorios que hay, únicamente, sobre el directorio de trabajo.

```
[usaurio@geniux ~]$ find . -type d|wc -l
```

3.6 HERRAMIENTAS PARA RESPALDO

El respaldo de información es una de las tareas básicas que debe relizar cualquier administrador. Es por ello que Linux ofrece algunas herramientas libres que nos permiten realizar esta labor.

He aquí algunas de estas herramientas:

3.6.1 LA HERRAMIENTA *cpio*

La herramienta cpio, fue creada para sustituir a la herramienta tar, que a su vez fue diseñada originalmente para almacenar archivos en cinta.

Formato:	cpio [opciones | grupo de opciones] < lista de archivos>archivo de respaldo.
Opciones:	-o Modo de salida.
	-i Modo de entrada.
	-v Mostrar mensajes.
	-d Crea directorios iniciales en donde se necesitan.

La función de esta herramienta es la de agrupar los de archivos en uno solo, para su posterior respaldo.

Se analizarán algunos casos para su mejor comprensión:

1.- Respaldar todos los archivos del directorio de trabajo actual.

En el caso de cpio, esta operación requiere de varios pasos:

Primero, habrá que guardar en un archivo de texto la lista de todos los archivos sobre el directorio a respaldar.

```
[usuario@geniux ~]$ find . - type f > lista.txt
```

Segundo, con el comando *cpio* se realizara el respaldo con la siguiente sintaxis:

```
[usuario@geniux ~]$ cpio -vo < lista.txt > respaldo.cpio
```

La sintaxis anterior puede resultar confusa, por ello se hará la aclaración de cómo se interpreta:

El contenido del archivo *lista.txt,* se envía como entrada estándar hacia la herramienta *cpio* (<) y el resultado del procesamiento bajo *cpio,* se redirecciona hacia el archivo *respaldo.cpio* (>).

Por último, es recomendable verificar que el archivo de respaldo exista y su tamaño de la siguiente manera:

```
[usuario@geniux ~]$ ls -lh respaldo.cpio
```

2.- Extraer la información almacenada en el archivo *respaldo.cpio.*

También el proceso de extracción se hará en varias etapas:

Primero, es recomendable crear un directorio para guardar los archivos al ser extraídos. Para este ejemplo se creará un directorio llamado *cpio-test* .

```
[usuario@geniux ~]$ mkdir cpio-test
```

En segundo lugar, copiar el archivo *respaldo.cpio,* hacia el directorio *cpio-test.*

```
[usuario@geniux ~]$ cp respaldo.cpio cpio-test/
```

Como tercera etapa, extraer los archivos contenidos en *respaldo.cpio.*

```
[usuario@geniux ~]$ cpio -ivd < respaldo.cpio
```

EJERCICIOS:

1.- Respaldar todos los archivos cuyo tamaño sea mayor a 1 kbyte, en un archivo llamado *mas1kbyte.cpio*.

3.6.2 LA HERRAMIENTA *tar*

Obtiene una copia de todos los archivos especificados en lista de archivos y los concatena en uno solo.

Formato:	tar [opciones | grupo de opciones] archivo-destino lista de archivos tar [opciones | grupo de opciones] archivo-destino directorio

Opciones:	-c	Agrupa .
	-x	Extrae .
	-v	Muestra la lista de archivos procesados.
	-f	Archivo o dispositivo destino.
	-r	Agregar al final del archivo.
	-t	Muestra la lista de archivo actualmente agrupados.
	-C	Especifica una ruta en donde van a ser desagrupados.
	-z	Comprime o descomprime los archivos agrupados en formato gz.
	-j	Comprime o descomprime los archivos agrupados en formato bz.

A pesar de existir substitutos, *tar* sigue siendo la herramienta más práctica y común utilizada para agrupar archivos. Para comprobar porqué, se analizarán los siguientes casos:

1.- Respaldar los archivos del directorio de trabajo cuya extensión sea **.txt,* y agruparlos en un archivo llamado t*exto.tar* .

```
[usuario@geniux ~]$ tar -cvf texto.tar *.txt
```

2.- Renombrar el archivo *telefono.txt,* como *telefono.txt.old,* y agregar el archivo *telefono.txt.old* al archivo *texto.tar* .

```
[usuario@geniux ~]$ tar -rf texto.tar telefono.txt.old
```

Para comprobar la inclusión del archivo, basta con ejecutar la herramienta *tar,* de la siguiente manera:

```
[usuario@geniux ~]$ tar -tf texto.tar
```

La opción -t, devuelve un resumen de los archivos contenidos dentro del archivo *texto.tar.*

3.- Normalmente, al respaldar, no únicamente se desea agrupar los archivos y guardarlos, además se busca que ocupen el menor espacio posible y para esto, la herramienta *tar,* permite agrupar y compactar archivos. Para lograr esto, se utiliza la opción -z, tal y como se muestra en el recuadro siguiente:

```
[usuario@geniux ~]$ tar -zcvf texto.tar.zp *.txt
```

Liste los archivos *texto.tar,* y *texto.tar.zp,* en su forma larga y observe la diferencia de tamaño entre ambos.

4.-verificar el contenido de un archivo compactado.

```
[usuario@geniux ~]$ tar -ztvf texto.tar.zp *.txt
```

Observe que se debe utilizar las mismas opciones acompañadas de una "z".

EJERCICIOS:

1.- Recuperar los archivos contenidos dentro del archivo *texto.tar.zp*.

3.6.3 LA HERRAMIENTA *gzip*

Es una aplicación que comprime archivos mediante el algoritmo Lempel-Ziv.

Formato:	gzip [opciones][lista de archivos]
Opciones:	-d Descomprimir
	-r Recorre el árbol de directorios recursivamente.
	-l Lista los atributos del archivo compreso.

Esta herramienta comprime el contenido de los archivos en el formato conocido popularmente como "zip".

1.- Comprimir el archivo *texto.tar,* verificar su tamaño y comparar el tamaño con el archivo texto.*tar.zp* .

```
[usuario@geniux ~]$ gzip texto.tar
[usuario@geniux ~]$ ls texto.tar* -lh
```

2.- Compactar todos los archivos con extensión **.txt,* en el directorio de trabajo.

```
[usuario@geniux ~]$ gzip *.txt
```

Esta acción "compacta" de manera individual todos y cada uno de los archivos con extensión "txt", sobre el directorio de trabajo.

3.- Descompactar todos los archivos con extensión *.tar.gz.

Para lograr esto, hay dos opciones que se muestran en el recuadro siguiente:

```
[usuario@geniux ~]$ gzip - d *.tar.gz
[usuario@geniux ~]$ gunzip *.tar.gz
```

La herramienta gunzip se puede utilizar para descompactar archivos "zip" aunque estos provengan de otro sistema operativo.

3.6.4 LA HERRAMIENTA *bzip2*

Es una aplicación que comprime archivos mediante un algoritmo mucho mas eficiente que el gzip.

Formato:	bzip2 [opciones][lista de archivos]
Opciones:	-d Descomprimir archivo.

Al igual que *gzip,* este comando comprime archivos o grupos de archivos pero con un algoritmo de compresión más poderoso.

Para comprender mejor como aprovechar mejor esta herraminta, es recomendable realizar los siguientes ejemplos.

1.- Copiar el archivo *telefono.txt* como *telefono2.txt,* utilizar *gzip,* para comprimir a *telefono.txt* y después, comprimir *telefono2.txt* con *bzip2,* por último comparar el tamaño entre ambos.

```
[usuario@geniux ~]$ gzip telefono.txt
[usuario@geniux ~]$ bzip2 telefono2.txt
[usuario@geniux ~]$ ls -lh telefono.*
```

2.- Descomprimir el archivo *telefono2.txt.bz2* .

```
[usuario@geniux ~]$ bzip2 -d telefono2.txt.bz2
```

3.6.5 HERRAMIENTA *split*

Permite partir un archivo en varias secciones.

Formato:	split [opciones][lista de archivos]
Opciones:	-b <num> Número de bytes a cortar.

Esta herramienta es muy útil cuando es necesario hacer respaldos en Floppy, Zip's o CDROM, y se da el caso que la información supera la capacidad del dispositivo de almacenamiento.

Los siguientes casos ilustran mejor cómo aprovechar esta herramienta:

1.- Crear un directorio que se llame *split-test,* copiar el archivo *telefono.txt,* hacia ese directorio y dentro de este, ejecutar lo siguiente:

```
[usuario@geniux split-test]$ split telefono.txt
```

Al listar en su forma larga el contenido de ese directorio, es posible ver que realizó una división arbitraria, tal como se muestra en el siguiente recuadro:

```
[usuario@geniux split-test]$ ls -lh
total 6.7M
-rwxr-xr-x    1 usuario  usuario   2.7M  nov    9 09:51      telefono.txt
-rw- r-- r--  1 usuario  usuario   123k  nov    9 09:54      xaa
-rw- r-- r--  1 usuario  usuario   123k  nov    9 09:54      xab
-rw- r-- r--  1 usuario  usuario   123k  nov    9 09:54      xac
...
```

Como puede observar *split,* no altera el archivo original, lo que hizo, fue crear un conjunto de pequeñas secciones.

Estos archivos fueron creados con un sufijo *(x**), pero,* ¿Por qué *split,* los cortó de ese tamaño?

La respuesta es muy sencilla, el comando *split,* por defecto corta el archivo en bloques de 1,000 líneas.

Este comportamiento puede ser útil, sin embargo es necesario tener un mejor control sobre su uso y para conocer como controlamos mejor el comportamiento de split, se presenta el siguiente ejemplo.

2.- Dividir el archivo *telefono.txt,* en bloques de 100k, y que el prefijo de los fragmentos sea *TELE.*

```
[usuario@geniux split-test]$ split --bytes=100k telefono.txt TELE
```

El resultado será:

```
[usuario@geniux split-test]$ ls -lh
total 6.6M
-rw- r-- r--  1  usuario  usuario   100k  nov  12   10:04   TELEaa
-rw- r-- r--  1  usuario  usuario   100k  nov  12   10:07   TELEab
-rw- r-- r--  1  usuario  usuario   100k  nov  12   10:07   TELEac
-rw- r-- r--  1  usuario  usuario   100k  nov  12   10:07   TELEad
...
-rwxr-xr-x  1  usuario  usuario   2.7M  nov  12     09:51     telefono.txt
```

Conociendo lo anterior, es posible dividir el archivo en fragmentos con el tamaño y el prefijo deseado, pero, ¿cómo es posible recuperar el archivo original a partir de los fragmentos?

El proceso es muy sencillo y se muestra en el siguiente recuadro.

```
[usuario@geniux split-test]$ cat TELE* > teles.txt
```

Para concluir con esta herramienta, sería bueno reflexionar lo siguiente :

¿Es posible hacer esto, con archivos de tipo binario?

La respuesta se deja como ejercicio.

3.7 HERRAMIENTAS PARA EL MANEJO DE PROCESOS.

Antes de poder continuar con la siguiente serie de herramientas, es necesario presentar algunos conceptos básicos sobre procesos.

Como se mencionó anteriormente, Linux es un sistema multiusuario y multitarea, el término multitarea, implica el manejo de múltiples procesos ejecutándose en memoria de forma aparentemente simultánea.

Definición de proceso: "*Un proceso es un programa en ejecución.*"

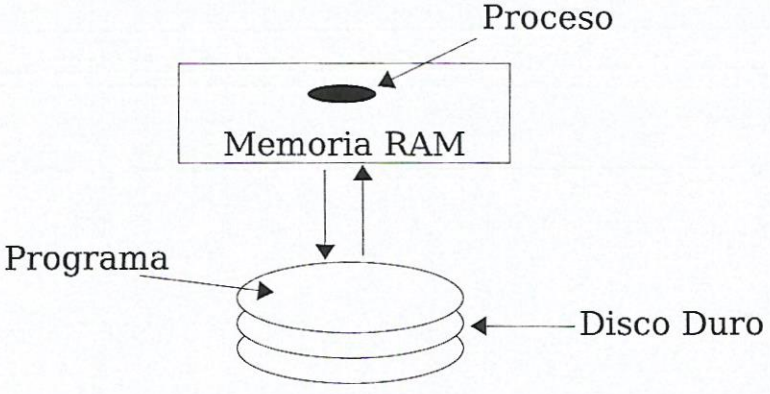


Figura 10. Esquematización de un proceso

Es muy importante que el usuario o administrador de un sistema tenga un control total sobre el comportamiento de los procesos que se ejecutan en el sistema a fin de evitar bloqueos o detectar aplicaciones maliciosas.

En Linux, existen varios tipos de procesos, entre los que destacan:

Proceso padre	Es un proceso a partir del cual se generan otros procesos.
Proceso hijo	Es un proceso creado a partir de otro proceso.
Proceso zombie	es un proceso que ha completado su ejecución pero aún tiene una entrada en la tabla de procesos.
Demonio	Son procesos que no tienen asociado ningún usuario ni tampoco una terminal específica.

3.7.1 LA HERRAMIENTA *ps*

Despliega la información del estado de los procesos activos.

Formato:	ps [opciones]	
Opciones:	-l	Muestra la información en su forma larga
	-e	Selecciona todos los procesos.
	-x	(todos los procesos)
	-a	Selecciona todos los procesos sobre una terminal
	-f	Genera un listado completo.

La herramienta *ps*, nos permite tomar una radiografía de los procesos o programas que se encuentran en ejecución en ese momento.

Para ilustrar esto, escriba lo siguiente:

```
[usuario@geniux ~]$ ps
```

Como verá, la información que nos devuelve es la siguiente:

```
 PID  TTY      TIME     CMD
1076  pts/1    00:00:00 bash
1095  pts/1    00:00:00 ps
```

La información del recuadro anterior corresponde a los procesos ejecutados en ese momento, en esa terminal.

La descripción de cada una de las columnas que aparecen, es la siguiente:

PID	Identificador de proceso.
TTY	Terminal desde donde se invocó el proceso.
TIME	Número de segundos en ejecución del proceso.
CMD	Nombre del comando

Ahora se tomará una radiografía más completa de lo que está sucediendo en el interior de su máquina, de la siguiente forma:

```
[usuario@geniux ~]$ ps -l
```

El resultado es una panorámica más completa, como puede apreciarse en el recuadro siguiente:

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	500	1076	1074	0	75	0	-	697	wait4	pts/1	00:00:00	bash
0	R	500	1096	1076	0	76	0	-	787	-	pts/1	00:00:00	ps

La información que presenta cada una de las columnas, es la siguiente:

PID Número de identificación del proceso.

TTY Numero de terminal que controla este proceso.

TIME Número de segundos que lleva en ejecución este proceso.

CMD Comando con el que se inició el proceso.

F Banderas.

001	ALIGNWARN	Impresión de un mensaje de alineación.
002	STARTING	Está siendo creado
004	EXITING	Se está eliminando
010	PTRACED	Se enciende si 'ptrace' (0), ha sido llamado.
020	TRACESYS	Trazado de llamadas al sistema.
040	FORKNOEXC	Bifurcado pero no en ejecución.
100	SUPERPRIV	Tiene privilegios de súper-usuario.
200	DUMPCORE	Vaciado de archivo *core*
400	SIGNALED	Eliminado por una señal.

Estado: El cual puede tener los siguientes valores:

- R Listo para ejecución.
- S durmiendo.
- D Letargo ininterrumpido (o prolongado).
- T Parado o trazado del proceso.
- Z para un proceso zombie.

UID Identificador de usuario.

PPID PID del proceso padre.

CPU % Utilización del procesador central.

PRI Prioridad

ADDR Dirección.

SZ Tamaño del proceso en bloques.

WCHAN Esta columna aparece en blanco para los procesos que están en ejecución.

A continuación se presenta una explicación mas amplia de las columnas que nos pudieran resultar mas útiles:

UID : Esta columna, identifica al usuario que está ejecutando a un proceso.

Cada uno de los usuarios,al ser dado de alta en en el sistema, contiene su UID, que se puede consultar en el archivo */etc/passwd*. La información sobre este archivo se analiza a fondo en el libro de esta serie titulado: *"Administración de Servidores Linux"*.

PID: Contiene el identificador de proceso con el que Linux, puede diferenciar cada uno de los procesos que se encuentran en memoria.

TTY: muestra la terminal a la que corresponde un proceso. No todos los procesos deben tener asociada una terminal. En el recuadro siguiente se muestran un grupo de procesos sin terminal.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	09:04	?	00:00:00	init [5]
root	2	1	0	09:04	?	00:00:00	[migration/0]
root	3	1	0	09:04	?	00:00:00	[ksoftirqd/0]
root	4	1	0	09:04	?	00:00:00	[watchdog/0]
root	5	1	0	09:04	?	00:00:00	[events/0]

Observe como la columna TTY, solo tiene un signo de interrogación.

3.7.2 LA HERRAMIENTA *pstree*

Muestra los procesos como un árbol; utiliza como raíz el PID, especificado o el proceso INIT, si este valor se omite y se recibe un nombre de usuario entonces, muestra todos los árboles cuyo proceso inicial pertenecen a este usuario.

Formato:	pstree [opciones] [pid] | [user]	
Opciones:	-p	Muestra el PID de cada proceso con un número decimal.
	-h	Muestra la ruta desde donde se ejecutó en el árbol de procesos.
	-a	Muestra si los procesos o comandos fueron ejecutados con parámetros.
	-H <PID>	(Hace lo mismo que -h pero este muestra la ruta del proceso PID listado).

Esta herramienta nos ofrece una forma sencilla de apreciar como se organizan los procesos dentro del sistema .

Para conocer mejor como funciona esta herramienta, ingrese al sistema como usuario *root,* y ejecute pstree, como se muestra en el recuadro siguiente:

```
[usuario@geniux ~]$ pstree
```

El resultado que debe obtener, es similar a lo siguiente:

```
init─┬─acpid
     ├─atd
     ├─auditd─┬─python
     │        └─{auditd}
     ├─automount───4*[{automount}]
     ├─avahi-daemon───avahi-daemon
     ├─bonobo-activati───{bonobo-activati}
     ├─crond
     ├─cupsd
     ...
     ...
     ├─6*[mingetty]
     ├─pcscd───2*[{pcscd}]
     ├─portmap
     ├─python
     ├─rpc.idmapd
     ├─rpc.statd
     ├─sdpd
     ├─2*[sendmail]
     ├─smartd
     ├─sshd
     ├─syslogd
```

```
      ├─udevd
      ├─watchdog/0
      ├─xfs
      └─yum-updatesd
```

Figura 11. Árbol de procesos.

Como se puede observar en la figura 11, este sistema tiene una estructura la cuál se contruye mediante una relación de procesos padres e hijos.

Observe que el proceso *init*, es el padre de todos los procesos, del y del cuál se deriva todos los procesos del sistema.

Una forma de apreciar el mismo árbol, y que nos permita ver el número de proceso (PID) es invocando la herramienta de la siguiente manera:

```
[usuario@geniux ~]$ pstree -ph
```

Un ejemplo de la salida se muestra en el recuadro siguiente:

```
init(1)─┬─acpid(2075)
        ├─atd(2222)
        ├─auditd(1806)─┬─python(1808)
        │              └─{auditd}(1807)
        ├─automount(2045)─┬─{automount}(2046)
        ...
```

El identificador de procesos se muestra a la derecha de cada uno del nombre del proceso y entre paréntesis.

3.7.3 LA HERRAMIENTA *jobs*

Esta herramienta tiene como única función tener una visión de los trabajos que se encuentran en el fondo del sistema.

Formato:	jobs.
Opciones:	No tiene opciones

Desde un punto de vista práctico, enviar un proceso al fondo, significa que al momento de ejecutarse, desocupe el *prompt,* (shell) del sistema, de tal suerte que es

posible poder ejecutar otras herramientas mientras la primera se encuentra en ejecución.

Los siguientes ejemplos ilustran mejor su funcionamiento:

Para ejemplificar esto, se enviará al fondo un proceso que busque todos los archivos cuyo nombre empiece con la letra 'a', y la salida de esta operación se enviará a un archivo llamado *salida.txt*, ubicado en el directorio de trabajo.

```
[usuario@geniux ~]$ find / -iname a* > salida.txt &
```

Para enviar un proceso al fondo, es necesario escribir al final de la línea el carácter '**&**'.

Al aplicar el comando *jobs*, la salida debe ser algo así:

```
[usuario@geniux ~]$ jobs
[1] Running        find / -iname a* > salida.txt &
```

Hay que tomar en cuenta que estos procesos, también por defecto envían su salida estándar a la pantalla y que la única diferencia es que desocupan el indicador del sistema (*prompt*), al realizar su trabajo en el "*fondo*".

3.7.4 LA HERRAMIENTA *fg*

Trae a primer plano el proceso enviado al fondo.

Formato: fg <num>

Opciones: Sin opciones.

En la herramienta anterior se ilustra cómo se pueden enviar los procesos al fondo y cómo poder listarlos mediante la herramienta *jobs*.

Ahora se presenta la herramienta *fg, que* permite traer a primer plano los procesos enviados al fondo. Para poder hacer esto se indica el número de proceso de acuerdo a la información presentada por la herramienta *jobs*.

Para comprobarlo envíe al fondo el mismo proceso, presentado en la herramienta *jobs*.

En el siguiente recuadro se muestra la información que envía a pantalla:

```
[usuario@geniux ~]$ find / -iname 'a*' > salida.txt &
[1] 1281
```

El número que nos interesa es el valor de la izquierda entre corchetes ([1]), este valor es el que se utiliza para regresar al proceso.

En el recuadro siguiente se muestra cómo se utiliza la herramienta *fg*, para lograrlo:

```
[usuario@geniux ~]$ fg 1
```

Para regresar el último proceso enviado al fondo, basta con invocar la herramienta *fg*, sin parámetros:

```
[user@geniux user]$ fg
```

3.7.5 LA HERRAMIENTA *kill*

Envía una señal a un proceso. Si no especifica alguna señal, kill, envía la señal TERM.

Formato:	kill [-señal] pid ... kill -l [señal]
Opciones:	-l Muestra una lista de las posible señales.

En su interior, Linux maneja un sistema de señales que permite al administrador controlar sus procesos, recuerde que en un sistema multiusuario pueden existir usuarios buenos y malos, que pueden ejecutar procesos buenos y malos.

Una de las razones por las que no es sencillo infectar de algún virus a un servidor Linux, es justamente porque un virus al ejecutarse o al esperar algún evento, se convierte en un proceso fácilmente localizable y eliminable.

kill, es como tener un arma en donde usted selecciona según la señal que aplicará a un proceso y utiliza el PID, para apuntar hacia un objetivo en especial (puede eliminar varios procesos de una sola ejecución, si así lo desea).

Suponga que se tiene la siguiente información:

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
100	S	504	1395	1176	0	60	0	-	587	read_c	tty1	00:00:00	bash
100	S	509	1588	1587	0	60	0	-	523	wait4	pts/2	00:00:00	su
100	S	509	1591	1588	0	71	0	-	580	wait4	pts/2	00:00:00	bash
000	S	509	1748	1321	0	61	0	-	547	do_sel	pts/1	00:00:00	vi
000	R	509	1752	1591	0	74	0	-	613	-	pts/2	00:00:00	ps

Para eliminar el proceso *vi,* con PID 1748, basta con ejecutar lo siguiente:

```
[usuario@geniux ~]$ kill -9 1784
```

Observe lo importante de tener a mano el PID, de cada proceso, con este es posible determinar exactamente cuál es el proceso o procesos a eliminar.

La opción -9 es la señal KILLTERM, esto quiere decir que estamos enviado la señal de terminar proceso al proceso con PID 1784.

En este sistema, cada vez que usted abre una terminal, todos los procesos que se crean a esta terminal (a través de invocar un comando en el shell), se subordinan al proceso de la terminal, que reconocen como proceso padre. En todas las terminales, este proceso padre es el proceso *bash*. Por lo tanto, si desea sacar a un usuario del sistema, busque y elimine el proceso *bash* que le corresponda.

EJERCICIOS:

1.- Abrir 2 sesiones de texto dentro del sistema, en la primera ejecutar el *vi,* y utilice la segunda, para eliminar dicho proceso.

2.- Abrir 2 sesiones de texto; utilice una de éstas para dar de baja la otra.

3.7.6 LA HERRAMIENTA *top*

Muestra una visión continua de la actividad del procesador en tiempo real.

Formato:	top [-] [d intervalo] [q]
Opciones:	-d Establece el periodo de actualizaciones en pantalla.
	-q Hace que top redibuje la pantalla sin intervalo alguno.
	-c Muestra la opciones de cada proceso.

Es una herramienta similar a *ps* con la diferencia de que en lugar de tomar una radiografía, toma una serie de muestras continuas cada 3 segundos, sobre el comportamiento de los procesos que consumen la mayor cantidad de recursos y las muestra en pantalla.

Por ejemplo, el siguiente recuadro muestra cómo ajustar la herramienta *top*, para actualizarse cada 2 segundos.

```
[user@geniux user]$ top -d2
```

La información que se envía a pantalla es la siguiente:.

```
top - 12:30:26 up  3:25,  0 users,  load average: 0.53, 0.45, 0.30
Tasks: 106 total,   2 running, 103 sleeping,   0 stopped,   1 zombie
Cpu(s):  5.9%us,  6.4%sy,  0.0%ni, 87.1%id,  0.0%wa,  0.5%hi,  0.0%si,  0.0%st
Mem:    223200k total,   210568k used,    12632k free,     3676k buffers
Swap:   522104k total,    55212k used,   466892k free,    96048k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2505 root      15   0 52340  11m 2640 S  9.0  5.2   8:31.81 Xorg
 2670 root      15   0 35752  13m  11m S  1.5  6.2   0:46.24 kicker
 2666 root      15   0 27148  11m  9m  S  0.5  5.2   0:19.20 kwin
 2674 root     -51   0 10288 3924 2840 S  0.5  1.8   0:04.83 artsd
 6897 root      15   0 34900  12m 9820 R  0.5  5.9   0:06.40 konsole
 7056 root      15   0 30440  14m 3512 S  0.5  6.6   0:30.76 wish
    1 root      15   0  2032  572  544 S  0.0  0.3   0:00.42 init
```

La información que envía a pantalla, se interpreta de la siguiente forma:

La primera línea, muestra entre otras cosas, la hora del sistema, el tiempo que ha transcurrido desde que el sistema se inicio, el número total de usuarios que tienen procesos en memoria y la carga promedio del sistema.

```
top - 12:30:26 up 3:25, 0 users, load average: 0.53, 0.45, 0.30
```

La segunda línea presenta el número total de procesos en el CPU, así como el número de procesos en estado de hibernación (durmiendo), corriendo, zombie o detenidos.

```
Tasks: 106 total,  2 running, 103 sleeping,  0 stopped,  1 zombie
```

La tercera línea permite cuantificar el porcentaje de CPU, utilizada para el usuario, sistema, porcentaje de procesos cuyos valores *nice* (el comando *nice* se explica mas adelante) son negativos y el porcentaje de procesos inactivos.

```
Cpu(s):  5.9%us,  6.4%sy,  0.0%ni, 87.1%id,  0.0%wa,  0.5%hi,  0.0%si,
0.0%st
```

La cuarta línea presenta la cantidad de memoria utilizada por el sistema.

```
Mem:  223200k total,  210568k used,  12632k free,   3676k buffers
```

La quinta línea presenta la información relativa a la memoria virtual.

```
Swap:  522104k total,  55212k used,  466892k free,  96048k cached
```

Después de la información general del sistema, la herramienta *top*, presenta un desglose de los procesos encabezando la lista a aquellos procesos que consumen la mayor cantidad de recursos.

Cada proceso en TOP, está dividido en columnas entre las que destacan:

PID	Identificador del proceso.
USER	El nombre del usuario que ejecuta ese proceso.
%CPU	Porcentaje de Procesamiento del CPU utilizado.
%MEM	Porcentaje de Memoria utilizada por el proceso.

Algunos son campos muy similares a las columnas presentadas por la herramienta *ps*.

3.7.7 LA HERRAMIENTA *nice*

Ejecuta un programa con la prioridad modificada.

Formato:	nice [-n ajuste] [comando [argumentos]]
Opciones:	-n prioridad a ajustar.

La herramienta nice, modifica la prioridad de los procesos, lo que permite tener procesos que se ejecuten mas rápidos que otros.

Si se pregunta porqué, la explicación es la siguiente:

Un proceso al ejecutarse entra en un cola, en donde espera su turno para que le presten el procesador durante un tiempo y pueda continuar la ejecución de su código. Al terminar este tiempo le pasa el procesador a otro proceso.

Si todos los procesos tienen la misma prioridad, el procesador pasa desde el inicio hasta el final de la cola uno detras de otro y no es posible tener el procesador hasta que no lo hayan utilizado todos los demas.

Al tener prioridades, habrán proceso que puedan tener mas tiempo el peocesador y eso permitirá que su código se ejecute por mas tiempo y por ello terminen mas rápido su ejecución.

En Linux, las prioridades van desde -20, hasta 19, siendo -20, el valor de mayor prioridad y el 19, el valor más bajo.

Cuando ejecute un *ps -l*, el valor *nice* (prioridad), se puede apreciar en la columna NI.

Para comrobar el uso de nice, primero, en un archivo llamado *test.sh,* hay que escribir lo siguiente:

```
#!/bin/bash
x=1
while [ $x != '0' ]
do
echo $RANDOM
done
```

Este archivo es un pequeño programa escrito para ejecutarse con el intérprete de comandos *Bash*.

Recuerde darle derechos de ejecución al programa, de lo contrario enviará un mensaje de error.

Por último, en dos sesiones de texto, ejecute el mismo programa, pero en una sesión ejecútelo en una prioridad alta y en otra sesión, con una prioridad más baja.

En el siguiente recuadro se muestra cómo ejecutarlo con la prioridad más baja:

```
[usuario@geniux ~]$ nice -n 19 ./test.sh
```

En el siguiente recuadro se muestra cómo ejecutar *./test.sh*, con una prioridad más alta:

```
[usuario@geniux ~]$ nice -n 0 ./test.sh
```

Debe observarse una diferencia perceptible en la velocidad de ejecución de ambos procesos.

Para detenerlos, utilice la combinación de teclas *Ctrl-C*.

3.7.8 LA HERRAMIENTA *init*

Inicialización en el control de procesos.

Formato:	init [0123456]
Opciones:	0 Apagar el sistema.
	1 Modo mono usuario.
	3 Modo multiusuario.
	5 Modo gráfico.
	6 reiniciar equipo.

Este comando, nos permite llevar al sistema a sus distintos estados de ejecución, de acuerdo a la configuración del archivo */etc/inittab*, que se estudiará en el libro de esta serie titulado: "*Administración de Servidores Linux.*"

La herramienta *init*, es muy importante a la hora de arrancar el sistema debido a que, según su configuración, son las opciones con las que inicia el sistema. A continuación una explicación rápida de cada uno de sus estados:

ESTADO 0: La herramienta *init*, lo interpreta como dar de baja todos los procesos del sistema y apagar el equipo.

ESTADO 1: El estado 1, es cuando el sistema se configura para que únicamente un solo usuario tenga el acceso al sistema. Este estado es muy útil cuando se desea dar mantenimiento a un servidor sin desconectarlo de la red.

ESTADO 3: Este estado significa levantar el sistema en modo texto y con capacidad de red para permitir que más de un usuario pueda hacer uso de sus servicios.

ESTADO 5: Este estado es similar al estado 3, con la diferencia que aquí se inicia el modo gráfico.

ESTADO 6: Este estado es para especificar el reinicio del sistema (*reboot*).

Para comprender mejor esto, ejecute el siguiente ejemplo:

1.-Arrancar el sistema en modo mono usuario.

```
[usuario@geniux ~]$ init 1
```

2.-Arrancar el sistema en modo multiusuario en texto.

```
[usuario@geniux ~]$ init 3
```

3.-Arrancar el sistema en modo multiusuario en gráfico.

```
[usuario@geniux ~]$ init 5
```

4.-Apagar el sistema.

```
[usuario@geniux ~]$ init 0
```

La configuración de cada uno de estos modos se puede modificar pero no corresponde al contenido de este libro explicar como.

Únicamente el usuario root tiene derechos para utilizar este comando.

3.8 HERRAMIENTAS DE RED.

Linux, como cualquiera de los sistemas operativos modernos, tiene herramientas que permiten comunicación entre los distintos usuarios del sistema, ya sea de forma local o remota.

El siguiente bloque, describe las algunas de estas herramientas y sus capacidades.

3.8.1 LA HERRAMIENTA *ssh*

Permite abrir una conexión hacia un servidor ssh.

Formato:	ssh usuario@servidor
Opciones:	No hay opciones.

Esta herramienta está diseñada bajo un esquema *cliente/servidor*, esto quiere decir que para poder establecer comunicación utilizando *ssh,* se necesita tener la aplicación servidor funcionando en el equipo destino.

Para levantar el servidor ssh consulte el **apéndice B.**

Estando el servidor listo es momento de hacer pruebas de comunicación y para ello considere realizar los siguientes ejemplos:

1.- Conectarse al servidor local de ssh.

```
[usuario@geniux ~]$ ssh 127.0.0.1
```

El protocolo ssh permite abrir sesiones de texto de manera remota. Así que por ello que manera natural, por lo menos debe recibir como parámetro la dirección IP del servidor a conectarse.

La primera vez que se conecta, hace la observación de que no puede establecer la identidad del host destino y pregunta que si desea continuar.

Para fines de este ejercicio la respuesta es "yes".

```
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
RSA key fingerprint is 43:38:8b:0c:b2:0d:8e:df:42:af:7e:0c:1f:00:4d:1f.
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '127.0.0.1' (RSA) to the list of known hosts.
```

Una vez superada esta etapa, ssh solicita el password del usuario root.

¿Porque el usuario root?

Si no se especifica usuario, ssh toma al usuario de la sesión actual como usuario de conexión.

```
root@127.0.0.1's password:
```

Si el nombre de usuario y contraseña, son correctos será posible conectarnos, de no ser así, el servidor enviará un mensaje de error, y rechazará la conexión.

Ya conectado se utiliza exactamente igual como cualquier sesión de texto.

2.- conectarse a un servidor ssh utilizando un usuario en específico.

```
[usuario@geniux ~]$ ssh usuario@127.0.0.1
```

3.8.2 LA HERRAMIENTA *who*

Despliega los nombres de los usuarios conectados actualmente en el sistema, además de sus números de terminal, la fecha y hora en que se dieron de alta.

Formato:	who[ami]
Opciones:	-H Imprime encabezado.
	-u Tiempo que el usuario ha estado ocioso.

La herramienta *who,* nos muestra qué usuarios están conectados al sistema vía una terminal (ya sea local o remota), y el número de terminal asignado.

Para comprobarlo escriba lo siguiente:

```
[usario@geniux ~]$ who
```

El resultado será similar a lo siguiente:

```
root     tty1     2007-11-12  14:49
geniux   tty2     2007-11-12  14:49
usuario  pts/4    2007-11-12  14:46  (Geniux)
```

En el recuadro anterior se ilustra cómo el usuario *root* está conectado en la primera terminal local (tty1), el usuario *geniux* en la segunda terminal local (tty2) y el usuario *usuario* en la terminal remota pts/4.

Si aún así siente que la información que devuelve es confusa, pruebe la siguiente variante:

```
[usuario@geniux ~]$ who -H
```

El resultado varía un poco:

```
NOMBRE   LÍNEA    TIEMPO          COMENTARIO
root     tty1     2007-11-12 14:49
geniux   tty2     2007-11-12 14:49
usuario  pts/4    2007-11-12 14:46  (Geniux)
```

Bueno, ahora haremos un cambio, invocando el comando de la siguiente manera:

```
[usuario@geniux ~]$ who -uH
```

El resultado ahora, es un poco más amplio. Analice el siguiente recuadro:

```
NOMBRE   LÍNEA    TIEMPO          INACTIVO   PID   COMENTARIO
root     tty1     2007-11-12 14:49  00:23    2503
geniux   tty2     2007-11-12 14:49  00:23    2504
usuario  pts/4    2007-11-12 14:46  00:08    2505  (Geniux)
```

La opción *–u*, devuelve el tiempo que ha permanecido inactiva la terminal. En la columna *INACTIVO* pueden aparecer cualesquiera de los siguientes valores:

hh:mm Tiempo en horas y minutos que lleva la terminal inactiva.
. El usuario ha estado activo el último minuto.
old El usuario ha estado inactivo por más de 24 horas.

Por último, suponga que pasa junto a un equipo con Linux y en consola de texto, el operador no se encuentra frente a esta y usted tiene la curiosidad de saber cuál es el usuario de sistema que tiene abierta esta terminal, para ello ejecute localmente la siguiente instrucción:

```
[usuario@geniux ~]$ whoami
```

El comando whoami devuelve el nombre del usuario que esta ejecutando la sesión que lo invoca.

3.8.3 LA HERRAMIENTA *mesg*

Habilita o deshabilita la recepción de mensajes enviados desde otros usuarios, mediante la utilidad write.

Cuando se llama a mesg, sin opciones informa si está habilitada o no.

Formato:	mesg [y/n]
Opciones:	No hay opciones.

En el sistema existe la herramienta *write,* que permite el envío de mensajes cortos entre usuarios que utilicen terminales de texto que, a diferencia de los mensajes de correo estos aparecen directamente en pantalla al momento de ser enviados. En algunas ocasiones estos mensajes de texto, pueden resultar incómodos u ofensivos. Por ello, esta herramienta habilita o deshabilita la recepción de mensajes.

Para comprobar por ejemplo, el estado de la recepción de mensajes en su terminal, escriba el siguiente mensaje.

```
[usuario@geniux ~]$ mesg
```

La respuesta puede ser o '*is y* ' o '*is n* ', dependiendo si está o no habilitada esta propiedad en su terminal de texto.

Para habilitar la recepción de mensajes, escriba lo siguiente:

```
[usuario@geniux ~]$ mesg y
```

Para desabilitar la recepción de mensajes, ejecute el comando y escriba como parámetro la letra 'n'.

3.8.4 LA HERRAMIENTA *write*

Dos usuarios pueden utilizar write, para establecer una comunicación básica.

Formato:	write usuario-destinatario [nombre-de-tty]
Opciones:	No hay opciones.

Esta herramienta, tal como se mencionó anteriormente, permite el envío de mensajes a distintos usuarios y dichos mensajes aparecen directamente en pantalla.

En el siguiente recuadro se presenta un mensaje al usuario *usuario*, conectado en la terminal remota No 2 y enviado por el usuario *geniux*.

```
[geniux@geniux ~]$ write usuario pts/2
mensaje  de texto
```

De esta manera el usuario *usuario,* recibirá el siguiente mensaje:

```
[usuario@geniux ~]$
Mensaje de geniux@geniux de pts/3 a las 11:59 ...
mensaje de texto
EOF
```

A partir de que se establece la comunicación, quien envía escribe líneas de texto y establece el final del mensaje hasta que presiona la secuencia de teclas Ctrl-D.

Cabe aclarar que tanto quien envía, como quien recibe el mensaje, debe tener la variable *mesg*, en 'y', para permitir la recepción de los mensajes.

3.8.5 LA HERRAMIENTA *mail*

Si no se especifican argumentos, lee los mensajes de correo del usuario que lo ejecuta. Con opciones, puede enviar mensajes a otros usuarios.

Formato:	mail [-s Subjet] [-c cc-addr] [to-addr]
Opciones:	No hay opciones.

Esta es una herramienta de comunicación para usuarios conectados a un mismo sistema vía una terminal remota o de texto, pero esta herramienta no funciona en tiempo real, es decir, la información llega a un buzón de correo y el usuario decide en qué momento consulta el mensaje. Esta herramienta también permite el envío de archivos.

Lo primero que se presentará, será el envío de mensajes y para ello, habrá que iniciar dos sesiones de texto, preferentemente con usuarios distintos.

Para los ejemplos que se presentan a continuación, se utilizan los usuarios *geniux* y *root*. El usuario root fue creado al instalar el sistema, pero el usuario geniux debe crearse con la siguiente secuencia de comandos:

```
[usuario@geniux ~]$ useradd geniux
```

La herramienta *useradd* permite crear una cuenta de usuario en el sistema.

```
[usuario@geniux ~]$passwd geniux
Changing password for user geniux.
New UNIX password:
BAD PASSWORD: it is based on a dictionary word
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

La herramienta *passwd* permite asignar o cambiar la contraseña a un usuario del sistema. El password debe escribirse dos veces y por seguridad, el sistema no refleja en pantalla lo escrito.

Para los ejercicios que se presentan a continuación se considera que se tienen 2 terminales de texto con el usuario root.

1.-Enviar un mensaje de correo del usuario *root* al usuario *geniux*.

```
[root@geniux ~]$ mail geniux <======= Destinatario.
Subject: mensaje de prueba      <======= Título del mensaje.
 esta es una prueba             <======= Mensaje.
.               <==== El mensaje debe concluir con un punto aislado
cc:             <====== Con copia para.
```

Como puede observar en el recuadro anterior, que en la línea de comandos se especifica el destinatario,con esta sintaxis, la herramienta sabe que se trata del envío de un mensaje de correo, entonces pregunta el título del mensaje y espera a que se escriba la información a enviar. Recordemos siempre que para finalizar un mensaje, se debe escribir un solo punto.

2.- Leer un mensaje de correo para el usuario root.

Para verificar si tiene correos nuevos en su cuenta de root, basta con escribir el comando *mail* sin opciones.

En caso de no tener mensajes aparecerá un mensaje similar al siguiente:

```
[root@geniux ~]$mail
No mail for root
```

Si su cuenta tiene mensajes de correo observará lo siguiente:

```
[root@geniux ~]$mail
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/root": 2 messages 2 new
>N  1 root@localhost.local  Sun Jun 28 14:54  13/613  "Llego un mensaje"
 N  2 logwatch@localhost.l  Sun Jun 28 14:54  42/1546  "Prueba de mail"
&
```

El recuadro anterior muestra la lista de los mensajes que se encuentran en el buzón.

Para leer el contenido del mensaje basta con escribir el número del mensaje en el prompt de la aplicación (&) y lo observará con el siguiente formato:

```
& 1
Message 1:
From root@localhost.localdomain  Sun Jun 28 14:54:34 2009
Date: Sun, 28 Jun 2009 14:53:39 -0500
From: root <root@localhost.localdomain>
To: geniux@localhost.localdomain
Subject: Llego un mensaje

mensaje  desde  el mismo
usuario  root
```

2.- Leer el buzón del usuario *geniux* desde *root*.

Aunque no es muy ético, es posible que desde root se puedan leer mensajes de otros usuarios invocando la herramienta mail de la siguiente manera.

```
[root@geniux ~]$ mail -f /var/spool/mail/geniux
```

La ruta /var/spool/mail/ es donde se guardan los "buzones" de correo en el sistema y solo el usuario root tiene acceso a todos los buzones.

3.- Borrar un mensaje.

Tomando como base el ejemplo anterior la información presentada en la herramienta mail puede ser la siguiente.

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/spool/mail/geniux": 2 messages 2 unread
>U  1 root@localhost.local  Sun Jun 28 14:54  18/665  "test mail"
 U  2 root@localhost.local  Sun Jun 28 14:54  19/693  "prueba de correo"
&
```

Para borrrar por ejemplo, el mensaje No 1 y 2 se escribe lo siguiente:

```
& d 1 2
```

La letra d nos dice que se van a borrar mensajes y los número 1 y 2 nos indican los mensaje a borrar.

3.9 HERRAMIENTAS MISCELÁNEAS.

Las herramientas que se presentan a continuación, no concuerdan con algunas de las categorías expuestas anteriormente pero no por ello dejan de ser importantes. Su conocimiento pudieran ser importantes para los administradores de servidores GNU/Linux, así que están invitados a conocerlas.

3.9.1 HERRAMIENTA *gcc*

La herramienta gcc, compila, ensambla y carga programas y fuentes escritos en lenguaje C.

Formato:	gcc [opciones] lista-de-archivos
Opciones:	- o (salida)

Éste, es el compilador de C ANSI, escrito para la GNU.

El lenguaje C, está profundamente asociado a UNIX®y sus derivados. Hay que recordar que en los Laboratorios Bell, a fines de los años 60', el mismo grupo de personas que inventó UNIX®, fue la gente que implementó las primeras versiones de C.

Si usted desconoce el manejo de lenguaje C, no importa, este ejemplo es únicamente para que tenga una idea del funcionamiento de la herramienta *gcc*.

Utilizando *vi*, escriba un archivo llamado *hola.c*, que contenga lo siguiente:

```
#include <stdio.h>
main()
{
  printf("Hola mundo \n");
}
```

Para generar un ejecutable binario a partir del archivo anterior , hay que escribir lo que se muestra en el recuadro siguiente.

```
[usuario@geniux ~]$ gcc hola.c -o hola
```

¿Qué hace la sintaxis anterior?

Se invoca la herramienta gcc, para leer el archivo *hola.c*, y de su contenido generar un ejecutable binario llamado hola.

El resultado será un pequeño programa que se ejecuta como se muestra en el recuadro siguiente:

```
[usuario@geniux ~]$ ./hola
```

3.9.2 HERRAMIENTA *cal*

Despliega el calendario de un mes o un año.

Formato:	cal [mes] año
Opciones:	No hay opciones

La herramienta *cal,* quizá por sí misma no tenga tampoco mucha utilidad, pero en ocasiones muy especiales, puede salvarle la vida. Su uso es muy sencillo como se muestra a continuación.

```
[usuario@geniux ~]$ cal
```

Por defecto muestra el calendario del mes en curso. Esto quiere decir que podemos consultar el calendario del mes en cualquier momento.

También es posible consultar fechas pasadas especiales ,por ejemplo si alguien desea saber el día de la semana nació, suponiendo que esta fecha es el 4 de Mayo de 1905 (observe el recuadro siguiente).

```
[usuario@geniux ~]$ cal 5 1905
```

En el ejemplo anterior puede apreciarse que el mes se especifica de manera numérica y que el año es con cuatro dígitos.

El resultado es el siguiente:

```
 mayo de 1905
do lu ma mi ju vi sá
    1  2   3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 1819 20
21 22 23 24 25 26 27
28 29 30 31
```

Ahora, supóngase que se desea tener el calendario del año 2009, y lo desea dentro de un archivo llamado *cal.txt*.

El procedimiento es el siguiente:

```
[usuario@geniux ~]$ cal 2009 > cal.txt
```

3.9.3 LA HERRAMIENTA *expr*

Evalúa una expresión y despliega el resultado.

Formato:	expr expresión
Opciones:	No hay opciones.

La herramienta *expr*, es de uso limitado debido a que evalúa expresiones lógicas y numéricas, y no permite el uso de valores de tipo flotante.

Sin embargo es muy sencilla de utilizar.

Las expresiones que evalúa esta herramienta puede contener los siguientes operadores:

*	Multiplicación	-	Resta
/	División	<	Menor que
%	Módulo	<=	Menor o igual
+	Suma	=	Igual
!=	Distinto de	>=	Mayor o igual

>	Mayor que	&	Operador AND
|	Operador OR		

Para comprender mejor su uso obsérvese la siguiente expresión:

```
[usuario@geniux ~]$ expr 2 + 3
```

El resultado es obviamente 5, sin embargo, al observar realmente esta herramienta se dará cuenta que, para que funcione correctamente se debe dejar un espacio entre los operadores y los operandos.

Cuando se utilicen paréntesis, estos deben ser precedidos por el carácter "\", de no ser así, serán interpretados de forma incorrecta. Para demostrarlo escriba y ejecute la siguiente expresión.

```
[usuario@geniux ~]$ expr \( 4 + 5 \) / 2
```

Los paréntesis, no pueden ser escritos de manera directa puesto que el intérprete de comandos los considera con un valor especial. Lo mismo sucede con la multiplicación, cuyo operador "*", como se recordará, se considera un comodín, por lo tanto, al hacer una multiplicación debe hacerla de la siguiente manera.

```
[usuario@geniux ~]$ expr 2 \* 3
```

Los siguientes ejercicios ilustran el uso de expresiones no numéricas.

```
[usuario@geniux ~]$ expr  4 != 5
```

La expresión anterior devuelve el valor de 1, debido a que su evaluación es verdadera.

```
[usuario@geniux ~]$ expr  4 != 4
```

Esta expresión devuelve el valor de cero, por lo tanto es falsa.

```
[usuario@geniux ~]$ expr \(  4 != 5  \)  \|  \(  8  !=  8 \)
```

La siguiente expresión en su conjunto es verdadera, puesto que la primera expresión (izquierda), es verdadera y se está utilizando el operador lógico OR ("|").

```
[user@geniux user]$ expr \( 4  !=  5  \) \& \( 8 != 8 \)
```

La expresión es falsa, puesto que ambas expresiones están separadas por el operador AND ("&").

3.9.4 HERRAMIENTA *man*

La herramienta *man*, es el manual que viene incorporado al sistema operativo, en él, es posible consultar los distintos parámetros de las herramientas del sistema, instrucciones de C, y archivos de configuración, entre otra información.

Formato:	man [sección] comando
Opciones:	No hay opciones.

Durante el recorrido de este libro, ha aprendido a manejar herramientas del sistema, pero solamente conoce algunas, y en el mejor de los casos, una subconjunto muy pequeño de sus parámetros.

Es difícil en un documento tan corto, estudiar todas los parámetros de cada herramienta, por ello, si se tiene la curiosidad de ver qué otras opciones ofrecen, sin duda las encontrará en el *man*.

El *man*, está dividido en ocho secciones, que se listan a continuación:

1. Herramientas de usuario.
2. Llamadas al sistema .
3. Subrutinas.
4. Dispositivos.
5. Formato de archivos.
6. Juegos.
7. Misceláneos.
8. Herramienta para el administrador.

La herramienta *man*, no únicamente contiene comandos, también contiene funciones de varios lenguajes y el contenido de algunos archivos de configuración, entre otra información.

Cuando se intenta localizar una palabra en *man,* es posible que el mismo término se encuentre en más de una sección, por lo que es recomendable en algunos casos, escribir la sección donde debe buscarla.

Por ejemplo, supóngase que está buscando la función *write* del lenguaje C, lo más lógico es buscar en el man, de la siguiente manera:

```
[usuario@geniux ~]$ man write
```

Al hacerlo, se observa que *man* devuelve información que no corresponde a lo buscado, en este caso las llamadas de lenguaje C, se encuentran en la sección 2, por lo tanto la búsqueda debe hacerse de la siguiente forma:

```
[usuario@geniux ~]$ man 2 write
```

Claro que se necesita un poco de práctica para tener un buen manejo de *man,* pero con el uso se vuelva más útil y sencillo de utilizar.

EJERCICIOS:

1.- Consulte en el manual del sistema el formato del archivo passwd.
2.- Consulte en el manual del sistema las opciones del comando ls.

3.9.5 HERRAMIENTA tty

Despliega en nombre de la terminal conectada a la salida estándar.

Formato:	tty
Opciones:	No hay opciones.

La herramienta *tty,* no hay mucho que agregar, recuerde que la utilizó anteriormente, para obtener el número de terminal que está usando.

Para las terminales gráficas el valor que devuelve puede ser el siguiente:

```
[usuario@geniux ~]$ tty
/dev/pts/2
```

En las terminales de texto, el valor devuelto pudiera ser el siguiente:

```
[usuario@geniux ~]$ tty
/dev/tty1
```

No olvide que una terminal está compuesta de tres dispositivos:

- ENTRADA ESTÁNDAR.
- SALIDA ESTÁNDAR.
- ERROR ESTÁNDAR.

Además recuerde también, que en Linux, existen dos tipos básicos de terminales:

/dev/ttyx (en donde x es el número de terminal): Terminales locales.

/dev/pts/x (en donde x es el número de terminal): Terminales remotas o gráficas.

3.9.6 LA HERRAMIENTA *mount*

Monta un sistema de archivos.

Formato:	mount *punto-de-montura*
Opciones:	-t tipo de sistema de archivos.

El comando *mount,* tiene la función de asociar una unidad de almacenamiento a un subdirectorio para que este sirva como punto de entrada y salida de información.

En el caso de la distribución utilizada en este curso (CentOS 5), monta automáticamente algunos dispositivos como son el *floppy* y la unidad de CD-ROM y las unidades USB, pero, no todas las distribuciones tienen esa facilidad o por ejemplo, al modificar los servicios de CentOS 5, pierde la capacidad de "automontaje".

Para comprender mejor el uso de esta herramienta veamos el siguiente ejemplo:

1.- Montar un diskette.

```
[usuario@geniux ~]$ mount /dev/fd0 -t msdos /mnt/floppy
```

Aquí, se pueden apreciar tres parámetros:

/dev/fd0	Archivo de dispositivo asociado a la unidad de diskettes.
-t msdos	La opción –t, sirve para especificar el tipo de sistema de archivos a montar. Entre los más populares tenemos:
	msdos Sistema de archivos utilizado por MS-DOS.
	iso9660 Sistema de archivos utilizado por los discos compactos de datos.
	ext2 Sistema de archivos utilizado por Linux.
	ext3 Sistema de archivos utilizado por Linux.
	auto Determina automáticamente el tipo de sistema de archivos.
/mnt/floppy	Directorio donde se asociará al dispositivo.

Es conveniente hacer la aclaración que cualquier directorio puede utilizarse para montar un dispositivo, pero, a fin de evitarse problemas, es recomendable crear un directorio específicamente para esta función.

Cada vez que no se ocupe una unidad que fue montada manualmente, es recomendable desmontarse manualmente antes de desconectar el dispositivo de almacenamiento.

Para el ejemplo anterior si se desea desmontar la unidad, se realiza la siguiente operación:

```
[usuario@geniux ~]$ umount  /mnt/floppy
```

Observe que para desmontar una unidad, únicamente se debe especificar el directorio asociado.

Los diskettes son dispositivos de almacenamiento obsoletos, pero el ejemplo anterior nos resulta util para comprender cómo funciona esta herramienta así que a continuación se presenta el mismo procedimiento para montar una unidad de almacenamiento USB.

2.- Montar una unidad USB.

El procedimiento es mas largo y se debe de tener conocimientos de algunas herramientas y archivos adicionales del sistema, sin embargo con la práctica se vuelve un procedimiento muy sencillo.

a) revisar el archivo */var/log/messages* con la herramienta *tail*.

```
[usuario@geniux ~]$ tail -f /var/log/messages
```

Como recordará la herramienta tail con su opción -f muestra la parte final de un archivo y actualiza la información en pantalla, conforme el archivo recibe la información.

b) insertar la unidad USB.

c) Buscar en el archivo */var/log/messages* la unidad asignada al dispositivo USB.

```
Jun 29 10:33:21 geniux kernel: sd 2:0:0:0: [sdb] Assuming drive cache: write through
Jun 29 10:33:21 geniux kernel: sdb: sdb1
Jun 29 10:33:21 geniux kernel: sd 2:0:0:0: [sdb] Attached SCSI removable disk
Jun 29 10:33:21 geniux kernel: sd 2:0:0:0: Attached scsi generic sg2 type 0
Jun 29 10:33:21 geniux kernel: usb-storage: device scan complete
Jun 29 10:33:22 geniux hald: mounted /dev/sdb1 on behalf of uid 0
```

El recuadro anterior muestra en negritas la información que nos dice cuál es el dispositivo asignado a la unidad.

d) Crear el directorio en el cuál se momtará el dispositivo.

```
[usuario@geniux ~]$ mkdir /media/dsk2
```

El directorio */media* es donde el sistema operativo establece por puntos de montura. En realidad se puede utilizar cualquier directorio pero a fin de llevar un orden es recomendable utilizar esta carpeta para este fin.

e) Montar la unidad.

```
[usuario@geniux ~]$ mount /dev/sdb1 -t auto /media/dsk2
```

Por último conjuntamos todos los elementos y montamos la unidad.

3.9.7 LA HERRAMIENTA *df*

La herramienta df, devuelve la cantidad de espacio disponible en los sistemas de archivos montados.

Formato:	df [opciones]
Opciones:	-a todos -h formato humano -m megabytes

Como se acaba de explicar, *mount* permite asociar los dispositivos de almacenamiento a un directorio, y a través de éste, tener acceso a su información, sin embargo, *mount* no especifica la cantidad de espacio disponible en cada dispositivo.

Esta sencilla herramienta nos devuelve la cantidad de espacio disponible en los dispositivos de almacenamiento montados. Al ejecutar *df*, sin parámetros devuelve algo similar a lo que se ilustra en el siguiente recuadro:

```
[usuario@geniux ~]$ df
S. ficheros     Bloques de 1K   Usado       Dispon      Uso%  Montado en
/dev/hda2       9920624         5520340     3888216     59%   /
/dev/sdb1       2011268         1025864     985404      52%   /media/KINGSTON
```

En el ejemplo anterior se tienen dos sistemas de archivos en activo, un disco duro (/dev/hda2) y una memoria usb (/dev/sdb1).

La información se muestra un poco confusa, los números que nos dicen la cantidad de espacio ocupado y espacio disponible no son muy claros por ello existe la opción *–h*, que devuelve la información en el siguiente formato:

```
[usuario@geniux ~]$ df -h
S. ficheros     Tamaño      Usado       Dispon  Uso%  Montado en
/dev/hda2       9,5G        5,3G        3,8G    59%   /
/dev/sdb1       2,0G        1002M       963M    52%   /media/KINGSTON
```

A éste, se le conoce como el formato "humano" y obviamente es mucho más sencillo de comprender.

3.9.8 LA HERRAMIENTA *du*

Estima la cantidad de espacio que ocupa un directorio o archivo.

Formato:	Formato: du [opciones] [archivo]
Opciones:	-a todos -b bytes -c total -h tamaño en formato leible por un humano.

La herramienta *du,* es muy útil cuando se desea saber el espacio que ocupa un directorio en el disco.

Si se aplica por ejemplo la herramienta *du,* al directorio */home/usuario,* esto se realiza de la siguiente forma:

```
[usuario@geniux ~]$ du /home/usuario
```

Devuelve la siguiente información:

```
4       /home/usuario/prueba4/etc/X11
8       /home/usuario/prueba4/etc
12      /home/usuario/prueba4
5600    /home/usuario/split-test
16      /home/usuario/.kde/Autostart
24      /home/usuario/.kde
4       /home/usuario/prueba3/bin
8       /home/usuario/prueba3
5704    /home/usuario
```

Esta información por sí sola no dice nada, pero si se utiliza la opción *–h,* el resultado es más comprensible:

```
4,0K    /home/usuario/prueba4/etc/X11
8,0K    /home/usuario/prueba4/etc
12,0K   /home/usuario/prueba4
5,5M    /home/usuario/split-test
16,0K   /home/usuario/.kde/Autostart
24,0K   /home/usuario/.kde
4,0K    /home/usuario/prueba3/bin
```

```
8,0K        /home/usuario/prueba3
5,6M        /home/usuario
```

Ahora tiene sentido la información devuelta y nos muestra el espacio que ocupa los archivos y directorios contenidos en este.

Como se mencionó anteriormente, existen ocasiones en las que sólo se necesita saber el espacio que ocupa un directorio en especial, ya sea para su respaldo o porque es necesario liberar espacio en el disco, etc.

Para conocer el espacio que ocupa, utilice las opciones *–hc,* y el resultado será lo siguiente:

```
[usuario@geniux ~]$ du -hc /home/usuario
4,0K        /home/usuario/prueba4/etc/X11
8,0K        /home/usuario/prueba4/etc
12,0K       /home/usuario/prueba4
5,5M        /home/usuario/split-test
16,0K       /home/usuario/.kde/Autostart
24,0K       /home/usuario/.kde
4,0K        /home/usuario/prueba3/bin
8,0K        /home/usuario/prueba3
5,6M        /home/usuario
5,6M  total
```

Observará información repetida, pero si la combinamos con otros comandos mediante un entubamiento (*pipe*), verá que puede resultar muy útil, por ejemplo, para quedarse únicamente con el total, puede hacerse de la siguiente manera:

```
[usuario@geniux ~]$ du -hc /home/usuario|grep total
```

EJERCICIOS:

1.- Investigar cuánto espacio ocupan los directorios: */etc, /bin, /home* y */usr/bin.*

3.9.9 LA HERRAMIENTA *which*

Muestra la ubicación de comandos que se encuentran en algunos de los directorios contenidos en la variable PATH.

Formato: which nombre-de-archivo

Opciones: No hay opciones.

En muchas ocasiones al instalar su distribución de Linux, no descarga todas las aplicaciones que desea, y debido a esto, se busca la herramienta para asegurarse que está instalada en disco, y una forma de hacerlo es utilizando la herramienta *find,* y la otra es utilizar which, de la siguiente manera.

```
[usuario@geniux ~]$ which rm
/bin/rm
```

La herramienta *which,* no busca las herramientas en todo el disco, sólo en aquellos directorios hacia donde apunta la variable PATH y toma en cuenta aquellos archivos que tienen derechos de ejecución.

Si tiene la curiosidad de saber en cuales directorios hace la búsqueda la herramienta *which,* ejecute lo siguiente:

```
[usuario@geniux ~]$ echo $PATH
```

Cada uno de los directorios están separados por el carácter ":" (dos puntos).

La variable PATH es considerada como una variable de ambiente y estas variables son utilizadas por los distintas herramientas y aplicaciones como parámetros implicitos.

Si desea conocer todas las variables de ambiente de su terminal, escriba la siguiente herramienta.

```
[usuario@geniux ~]$ env
```

3.9.10 LA HERRAMIENTA *whereis*

Localiza el ejecutable binario, el código fuente y el manual para una herramienta.

Formato: whereis [opciones] nombre-de-programa
Opciones: -b binario -m manual -s código fuente

Para fines prácticos, puede utilizarse en lugar de la herramienta *which,* aunque tienen contextos distintos.

Por ejemplo para la herramienta *rm* regresa lo siguiente:

```
[usuario@geniux ~]$ whereis rm
rm: /bin/rm /usr/share/man/man1p/rm.1p.gz /usr/share/man/man1/rm.1.gz
```

EJERCICIOS:

1- Utilizando esta herramienta, observe la información que se devuelve de las siguientes herramientas: *apache, perl, grub* y *open.*

3.9.11 LA HERRAMIENTA bc

La herramienta bc, es un lenguaje que soporta números de precisión arbitraria con ejecución interactiva de comandos. Este lenguaje tiene una sintaxis similar al lenguaje C.

Formato: bc [opciones]
Opciones: -i Forza al modo interactivo. -l Define la libreria matemática a utilizar.

bc, corrige todas las deficiencias que tiene el comando *expr,* este último es simplemente un evaluador de expresiones mientras que *bc,* es todo un lenguaje para programación.

Observe cómo la definición de *bc* no incluye una expresión, esto se debe a que *bc,* es una herramienta para uso en modo interactivo, algo similar a un intérprete, sin embargo, es posible que reciba y evalue expresiones simples, de la siguiente manera:

```
[usuario@geniux ~]$ echo "4 + 5"|bc
```

El comando *echo,* despliega la operación y *bc,* la evalúa. Además, corrige muchos de los problemas que tiene la herramienta *expr*, como son:

- Son innecesarios los espacios en blanco entre operandos y operadores.
- Es innecesario anteponer el carácter "\", antes de paréntesis o el signo "*".
- Maneja cantidades con precisión arbitraria.

Para comprobar lo anterior, basta con escribir la siguiente expresión:

```
[user@geniux user]$ echo "4.25/4"|bc
```

Esta expresión devuelve un resultado equivocado, porque el número de decimales que entrega como resultado, por defecto es cero. Para corregir esto basta con hacer la siguiente modificación:

```
[user@geniux user]$ echo 'scale=5;4.25/4'|bc
```

La palabra reservada *scale,* nos dice el número de decimales despúes del punto decimal que debe entregar.

También es posible utilizar un archivo de texto y hacer un pequeño programa para *bc.* Para comprobarlo elabore un archivo llamado *numeros.bc,* que contenga lo siguiente:

```
#!/usr/bin/bc
14+3
5*2
17%2
scale=4
8.2 *3.25
quit
```

Este pequeño archivo para ejecutarlo asigne derechos de ejecución (*chmod*), y se ejecuta como cualquier programa (*./numeros.bc*).

El resultado debe ser el siguiente:

```
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
17
10
1
26.650
```

Cae fuera de los objetivos de este libro, desarrollar un curso completo de la herramienta *bc*; si desea mayor información sobre el tema puede consultar el manual (*man*), que contiene una descripción muy completa de todas las funcionalidades de este intérprete.

APÉNDICE A

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in

this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all

its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may

be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been

modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR

INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify
> it under the terms of the GNU General Public License as published by
> the Free Software Foundation, either version 3 of the License, or
> (at your option) any later version.
>
> This program is distributed in the hope that it will be useful,
> but WITHOUT ANY WARRANTY; without even the implied warranty of
> MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
> GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License
> along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.
If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

APÉNDICE B

EL SERVICIO SSH.

Levantar el servicio ssh es muy sencillo, y el procedimiento para hacerlo lo podemos resumir en los siguientes pasos:

1.- Verificar que el servidor ssh está instalado.

```
[root@geniux ~]# rpm -q openssh
```

Para estar seguros de que el servidor ssh está instalado, la herramienta *rpm* , debe devolver el nombre y la versión del paquete tal y como se muestra a continuación.

```
openssh-4.3p2-16.el5
```

En caso contrario debe devolver el siguiente mensaje:

```
package openssh is not installed
```

2.- Instalar el paquete openssh

La instalación de un paquete es un procedimiento sencillo, sin embargo antes de de realizar esta operación hay que "montar" el dvd de CentOS 5.2 de la siguiente manera:

a) Insertar el disco de CentOS 5.2.

b) Esperar el auto montaje desde el ambiente gráfico o montarlo manualmente desde el ambiente texto utilizando la herramienta mount (ver sección 3.9.6).

c) Cambiarse a la carpeta en donde se encuentran los paquetes rpm de instalación.

```
[root@geniux ~]# cd /media/Carpeta_De_Montaje/CentOS/
```

d) Instalar el servidor ssh

```
[root@geniux ~]# rpm -i openssh-server-4.3p2-26.el5.i386.rpm
```

e) Instalar el cliente ssh.

```
[root@geniux ~]#rpm -i openssh-clients-4.3p2-26.el5.i386.rpm
```

3.- Modificar el archivo /etc/ssh/sshd_config.

El archivo sshd_config configura el "demonio" del ssh y la opción que nos interesa se muestra en el siguiente recuadro:

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
```

La opción *PermitRootLogin* permite que se puedan conectar a un servidor ssh como usuario root, así que debe borrarle el carácter "#".

No utilice esta opción para servidores en producción , es muy inseguro.

4.- Levantar el servicio ssh.

```
[usuario@geniux ~]$ service sshd start
```

Índice alfabético

Esta obra es el resultado de mas de 10 años de experiencia en capacitación en GNU/Linux por parte de Geniux Consultoría y Capacitación.

Contiene el uso de las 50 herramientas de consola de texto mas comunes para el manejo de GNU/Linux, también ofrece un método muy sencillo para su aprendizaje y a diferencia de otros textos contiene una explicación concreta de las filosofías del software libre.

El documento se estructura en 3 bloques: El primero contiene un resumen muy concreto acerca de la filosofía de software libre y open source, el segundo explica los origenes de UNIX y los fundamentos de las consolas de texto, el tercer bloque explica el uso de los comandos en texto mas comunes en GNU/Linux.

Daniel Enrique Vázquez Solís titulado en *Ingeniería en Sistemas Electrónicos*, por el Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Cuernavaca.

Desde 1997 se dedica al estudio y promoción del sistema operativo GNU/Linux. Ha desarrollado su labor de consultoría para los sectores empresarial, gobierno e instituciones de educación superior.

Actualmente es director general de **Geniux Consultoría y Capacitación**, despacho dedicado al desarrollo de soluciones en tecnologías de Información y capacitación en **GNU/Linux**, además es profesor en el Instituto Tecnológico de Acapulco.

Geniux Press

www.geniux-online.com

ISBN 978-970-95789-0-4

9 789709 578904