Administración de Linux I

Manual de referencia

Índice del Manual Administración de Linux I 2009.0

1 . Presentación

1.1 . Sobre Fortix

Fortix es una empresa pionera en el uso de Software Libre en el Noroeste Argentino que cuenta en su plantel de profesionales con expertos en el software libre y sistemas abiertos.

Entre nuestros servicios brindamos consultaría en informática, seguridad informática, cursos, instalación de servidores Linux para Internet y LANs, migración de sistemas en plataformas comerciales a plataformas de software libre, desarrollo de sistemas a medida y otros. Para más detalles no dude en consultarnos.

Dirección: Avenida Sarmiento 1347 - Oficina B
Ubicación: (T4000) San Miguel de Tucumán, Tucumán, República Argentina
Teléfono: 00 54 381 423 7797
Web Site: http://www.fortix.com.ar
E-mail: contact@fortix.com.ar

1.2 . Sobre este manual

Fortix y los autores de este manual no son responsables de cualquier daño que pueda ser ocasionado por el uso o mal uso del contenido del mismo. **Fortix** se reserva el derecho de modificar el contenido de las próximas ediciones de este manual sin previo aviso a los usuarios.

Aviso legal

2 . Conceptos básicos

2.1 . Breve introducción técnica

Antes de comenzar, hay aclaraciones de mucha relevancia que hay que hacer respecto a los sistemas Linux y al lenguaje en este manual:

- En los sistemas Linux (y UNIX en general) sí importan las mayúsculas y minúsculas (case sensitive). Al escribir un comando, un patrón de búsqueda, expresión regular, etc. **se debe respetar mayúsculas y minúsculas**.
- En Linux los archivos no tienen algo así como un *nombre* y una *extensión* (como en los sistemas Windows), sino que tienen **nombre** (lo que se ve cuando se listan los archivos en un directorio) y una **finalización**, que indica qué tipo de archivo es y que se encuentra contenida en él. El nombre de archivo puede finalizar en *.jpg* y ser un archivo de texto plano.
- La filosofía UNIX (y por herencia Linux) es trabajar siempre sobre archivos: todo en el sistema se trata de archivos. Incluso **los directorios son archivos**. Esto tiene ventajas impensables por ahora.
- Cuando se habla de *demonios* se hace referencia a servicios.
- No existe algo así como un atributo de archivo que indique que es un archivo oculto, sino que se usa la convención de que los archivos cuyos nombres comienzan con un punto (".") son archivos ocultos.
- Unidades de almacenamiento de memoria:

bit(b) = un dato (un 0 ó un 1)
Byte(B) = 2^3b
KiloByte(KB) = 2^{10}B = 2^{13}b
MegaByte(MB) = 2^{10}KB = 2^{20}B = 2^{23}b
GigaByte(GB) = 2^{10}MB = 2^{20}KB = 2^{30}B = 2^{33}b
TeraByte(TB) = 2^{10}GB = 2^{20}MB = 2^{30}KB = 2^{40}B = 2^{43}b
PetaByte(PB) = 2^{10}TB = 2^{20}GB = 2^{30}MB = 2^{40}KB = 2^{50}B = 2^{43}b
ExaByte(EB) = 2^{10}PB = 2^{20}TB = 2^{30}GB = 2^{40}MB = 2^{50}KB = 2^{60}B = 2^{43}b
ZettaByte(ZT) = 2^{10}EB = 2^{20}PB = 2^{30}TB = 2^{40}GB = 2^{50}MB = 2^{60}KB = 2^{70}B = 2^{43}b
YottaByte(YT) = 2^{10}ZB = 2^{20}EB = 2^{30}PB = 2^{40}TB = 2^{50}GB = 2^{60}MB = 2^{70}KB = 2^{80}B = 2^{43}b

Hay desacuerdo principalmente con respecto a ZettaByte y YottaByte, pues algunos se refieren a ellos como 2^{70}B y 2^{80}B, respectivamente, mientras que otros les atribuyen 10^{21}B y 10^{24}B. Para salvar la ambigüedad se habla de ZobiByte (ZiB) y YobiByte (YiB), que se expresan obligadamente en potencias de dos como aparece en la tabla. Así también, se puede hablar de ExbiByte (EiB), PebiByte (PiB), TebiByte (TiB), etc. Para saber más puede investigar en Internet las frases claves "prefijos binarios", "Norma CEI", "IEEE 1541". En este manual usaremos la notación dada en la tabla de arriba.

2.2 . Distribuciones

2.2.1 . ¿Qué son las distribuciones?

Linux y todo el software libre se crea, desarrolla y distribuye por Internet.

Una distribución es una recopilación de software para Linux hecha por alguna organización o por particulares, generalmente con alguna orientación particular como servidor, estación de trabajo, edición de imagen, video, sonido, etc. o puede ser de propósito general. Actualmente la forma más popular de instalar Linux es con una distribución ya que sería muy difícil armar un sistema Linux desde cero.

Las empresas que se dedican al armado de distribuciones cobran por la recopilación del software, no por el software en sí, y por el soporte técnico multilingüe impreso, aunque también existen distribuciones que son absolutamente gratuitas y son mantenidas por la comunidad internacional de usuarios y desarrolladores de la misma (tal es el caso de Debian). La tarea de quienes se dedican a mantener y desarrollar distribuciones es la de recolectar software libre, armar un instalador para la distribución y hasta brindar soporte en algunos casos. También el soporte se puede conseguir en los foros de Internet y en el foro oficial de la distribución que estemos usando, aunque no tenemos garantía de conseguir la mejor respuesta (ni siquiera de obtener una), por lo que existen profesionales y expertos en Linux dispuestos a brindar soluciones por un precio módico. Tal es el caso de **Fortix**.

Otra tarea que enfrentan los desarrolladores de las distribuciones es adaptar el software libre a su distribución. Es muy importante saber que los desarrolladores de software libre en general sólo liberan el código fuente en Internet (para que cualquier persona o entidad disponga de él libremente bajo la licencia a la que está sujeta) y cada distribución se encarga de preparar los instaladores, dependencias, etc. Visto de otra forma: Apache es Apache, no Ubuntu, no Gentooo, Apache es un servidor de Internet que puede correr sobre muchas plataformas.

Algo para tener en cuenta es que en el contexto de Linux (foros y demás) se suele llamar “distros” a las distribuciones como una forma más familiar de referirse a ellas.

2.2.2 . Ejemplos de distribuciones

openSUSE: Distribución muy popular. Última versión: 11.1. De propósito general: desde estación de trabajo hasta servidor. Esta distribución tiene muchos desarrollos propios, como software de instalación, administración, esquemas de seguridad, etc.

redhat: Distribución de origen norteamericano, la más usada del mundo. Tiene muchos desarrollos propios también, como el formato de paquetes rpm ampliamente usado por otras distribuciones.

Debian: No tiene origen, es creada por programadores de todo el mundo. Es la distribución GNU por excelencia, ya que su desarrollo no persigue fines economicos. La última versión liberada es 4.0-r7.

Gentoo (“yen-tú”): Esta distribución tiene una característica que la hace muy especial: se autogenera desde cero. Con ella podemos tener un sistema completamente compilado en nuestra computadora, lo cual brinda una optimización y performance incomparables a la par de las otras distribuciones en las cuales los binarios son compilados por los desarrolladores de la distribución. Otra característica especial es su sistema de gestión de paquetes (reminiscencia del modelo de ports de BSD) llamado Portage. Su última versión es Gentoo 2008.0-r8.

2.3 . Conceptos básicos de particionamiento de discos

En los sistemas Linux actuales (y, en general, en los sistemas UNIX) un disco rígido puede tener como máximo cuatro particiones primarias, de las cuales puede haber una que sea extendida, significando ésto que puede contener en sí misma una o más particiones lógicas, llegando a un máximo de 64 particiones.

Las particiones extendidas no contienen datos en forma directa, sino que son las particiones lógicas dentro de ellas las que los contienen.

Usualmente se usan diferentes particiones para facilitar transiciones entre sistemas operativos o para tener más de uno funcionando dentro de un mismo disco rígido. Cada partición lógica o primaria puede contener un sistema operativo y pueden ser formateadas con el sistema de archivos que se crea conveniente. La ventaja de usar diferentes particiones es que se puede usar una para almacenar el sistema operativo y los programas, otra para documentos y otra más para guardar archivos descargados de Internet. De esa forma se asegura un mínimo de seguridad de tal forma que se puede borrar el contenido de la partición del sistema operativo y los programas e instalar otro sistema operativo sin tener que hacer back up's de los datos en la partición de documentos. También puede ocurrir que el sistema de archivos de la partición de programas se dañe al haberse interrumpido la alimentación del disco mientras se escribía en él, entonces al tener particionado el disco será solamente el sistema de archivos de la partición de programas el que se haya dañado y no el de la partición de documentos (con suerte).

2.4 . Árbol de directorios en Sistemas UNIX y montaje de dispositivos

En los sistemas operativos Microsoft Windows los dispositivos de almacenamientos que están conectados al equipo y las particiones primarias y lógicas se listan como árboles de directorios separados (no conexos), cada uno con una letra de la A a la Z que los identifican (cada letra hace referencia a un dispositivo a lo sumo), y son llamados unidades.

Para el caso, puede entenderse como dispositivo de almacenamiento a cualquier medio de almacenamiento como pendrive, tarjeta de memoria, disquette, partición primaria o lógica de un disco rígido, o a un disco rígido sin particionar, siempre que tenga un sistema de archivos reconocible por el sistema operativo.

Si se inserta un dispositivo de almacenamiento removible como un pendrive en Windows es reconocido y tan pronto como sea posible se dispone de una nueva unidad (una letra que no

estaba asignada) de donde se desprende el árbol de directorios del dispositivo de almacenamiento. Antes de retirar el pendrive, se procede a la "Extracción segura" del dispositivo a través de una interfaz gráfica (generalmente). La ejecución de esta orden no hace otra cosa que detener la lectura y escritura en el dispositivo (si es posible) para que la extracción no provoque corrupción en los datos al escribirlos a medias, entre otros daños.

En los sistemas tipo UNIX todos los dispositivos de almacenamiento conectados al equipo están listados en un directorio especial. No existen las unidades al modo Windows. El sistema entero se puede ver como un sólo árbol de directorios (no como una colección de ellos —árbol de directorios de la unidad **c:**, árbol de directorios de la unidad **d:**, etc.), completamente conexo, su raíz se representa por **/** (barra de división normal) y se conoce como directorio raíz, y el separador de directorios es la misma barra. Un ejemplo de directorio puede ser **/home/usuario/**, en vez de **f:\Mis documentos\usuario**.

Entonces, ¿qué pasa cuando se conecta un pendrive y se quiere ver la información, el árbol de directorios contenido en él? Para hacerlo es necesario **montar** el dispositivo en alguna carpeta cualquiera dentro del árbol de directorios raíz (en cualquier nivel del árbol). A partir de entonces, lo que haya contenido esa carpeta ya no es visible, en cambio, se puede visualizar todo el árbol de directorios del pendrive.

Si en el directorio **/mnt/pendrive/** (que sólo contiene un archivo llamado **ejemplo**) montamos un pendrive que contiene un sólo archivo llamado **ejemplo1** sólo se podrá visualizar **/mnt/pendrive/ejemplo1** y no **/mnt/pendrive/ejemplo**.

Cuando nos dispongamos a retirar el pendrive tendremos que **desmontarlo** (esto en Windows es la "Extracción segura").

El directorio /mnt/pendrive/ es el PUNTO DE MONTAJE del dispositivo.

Con frecuencia, Linux no monta en forma automática los dispositivos de almacenamiento (salvo que lo explicitemos, se lo puede hacer con varios programas) por razones de seguridad y por otros motivos. En Linux, podemos tener cuatro particiones primarias en el mismo disco rígido y tener montadas solamente tres, ya que tal vez a una no la vamos a usar.

El montaje de dispositivos en los sistemas UNIX tiene muchísimas ventajas que, a lo largo de este curso, se van a hacer evidentes.

3 . Instalación

3.1 . Cómo conseguir Linux

Conseguir Linux es muy fácil. Se lo puede bajar de Internet, pedir prestado a un amigo, de CD's que vienen con revistas, etc. También se puede bajar el núcleo de Linux (en adelante kernel) del sitio oficial **http://www.kernel.org** y los programas de **http://freshmeat.net**, **http://sourceforge.net/** o **http://rufus.w3.org** y bajar el resto del software (como el gestor de paquetes y los paquetes básicos) de alguna distribución.

Si no estamos interesados en armar un sistema Linux desde cero podemos recurrir a las distribuciones: podemos comprar una (los precios son realmente bajos si los comparamos con otro software comercial, además de contar con soporte técnico, manuales completos en nuestro idioma, etc.), podemos copiarla si la conseguimos prestada o también podemos descargar de Internet la(s) imagen(es) ISO de CD de la distribución de nuestra elección y quemar los CD's correspondientes, siendo todo esto totalmente legal. Se recomienda siempre conseguir la versión más reciente de la distribución, por muchas razones: mayor compatibilidad de hardware, más funcionalidades y porque los problemas de las versiones anteriores ya habrán sido superados. Si el equipo que estamos por instalar estará expuesto, como un servidor de Internet, conseguir la última versión de una distribución es una excelente política de seguridad para poner en práctica.

Para descargar la imagen ISO de una distribución lo habitual es dirigirse a su sitio web oficial. A continuación, los sitios oficiales de las distribuciones más populares:

```
http://www.gentoo.org
http://www.debian.org
http://www.suse.com
http://www.redhat.com
http://www.slackware.com
http://www.linux-mandrake.com
```

3.2 . Antes de descargar

En Internet, cuando se está por bajar la imagen ISO de alguna distribución, es muy común encontrar muchas opciones:

Plataforma: x86, sparc, amd64, i686, i386, etc. Se refiere a la arquitectura del microprocesador de la computadora sobre la que va a correr nuestro Linux.

Server / Workstation / Desktop: cuál es la orientación, el propósito, el uso que le vamos a dar. En esta categoría pueden aparecer más opciones.

LiveCD / InstallCD: determina el propósito inmediato del CD (o DVD) que descargamos. InstallCD significa que sólo se dispondrán de las herramientas suficientes para la instalación de la distribución completa. LiveCD ocupa más espacio, a costa de permitir correr el sistema operativo Linux con la distribución **completos desde** el CD. Es una herramienta muy útil si es que no estamos seguros si queremos o no ese sistema instalado definitivamente en nuestro

equipo. El kernel de Linux se carga en la RAM así como toda aplicación que se use, lo que implica una velocidad de trabajo inferior a la que se obtendría con la instalación en disco. En esta categoría de opciones puede haber más que las que mencionamos, dependiendo de la distribución.

Dependiendo de la distribución, puede haber más opciones, pero éstas son las más importantes.

3.3 . Recopilar información de hardware

La gran mayoría de las distribuciones orientadas a usuarios de escritorio hoy en día detectan con mucha precisión el hardware que poseemos. Sin embargo, es recomendable conocer bien nuestro hardware para pulir detalles manualmente (especialmente si estamos instalando un servidor, donde la optimización es más importante).

Es importante saber que en Linux para configurar un hardware lo que necesitamos saber es el chipset y no la marca del mismo.

Algunas de las cosas que tenemos que tener en cuenta sobre nuestro hardware:

- Marca, modelo, chipset y cantidad de memoria de la placa de viedeo.
- Marca, modelo y chipset de la placa de sonido.
- Marca, modelo y frecuencias del monitor.
- Marca y modelo del módem.
- Sobre los discos rígidos: tipo, capacidad y tabla de particiones.
- Tipo de unidad de CD-ROM.

Además también tendremos que especificar la distribución del teclado (E.E.U.U., Internacional, Español Internacional, etc.) y el tipo de mouse que tenemos (si la ficha es microdim, estilo PS/2, tipo DB9 hembra, mouse serial, en cuyo caso tendríamos que saber si está conectado al serial 1 ó 2). Lo normal es que Linux encuentre automáticamente el mouse, en cuyo caso no hace falta configurarlo.

3.4 . Nombres de dispositivos de almacenamiento

Como se había anticipado, todos los dispositivos (no sólo los de almacenamiento) están listados en forma de archivos en un directorio especial del sistema. Este directorio es **/dev/**.

Comenzaremos suponiendo una PC con dos bancos IDE: primario y secundario (recordemos que cada uno puede tener adjuntos dos dispositivos: el maestro y el esclavo). Los nombres de los dispositivos IDE en Linux son entonces:

Banco	Conexión	Dispositivo
IDE0	Primario Maestro	/dev/hda
IDE0	Primario Esclavo	/dev/hdb
IDE1	Secundario Maestro	/dev/hdc
IDE1	Secundario Esclavo	/dev/hdd

Tabla 1: Nombres de dispositivos IDE

Ejemplo: tenemos conectado un disco rígido IDE como Primario Maestro y una lectora de CD como Primario Esclavo, sus nombres son /dev/hda y /dev/hdb, respectivamente.

Si los dispositivos son SATA o SCSI en vez de IDE, los nombres son:

Disco	Conexión	Dispositivo
0	Primer disco SCSI o SATA	/dev/sda
1	Segundo disco SCSI o SATA	/dev/sdb
2	Tercer disco SCSI o SATA	/dev/sdc
3	Cuarto disco SCSI o SATA	/dev/sdd
4	Quinto disco SCSI o SATA	/dev/sde

Tabla 2: Nombre de dispositivos SATA / SCSI

Cada disco debe tener al menos una partición para poder trabajar con ella. En linux, cada partición de cada disco tiene asociado, además, otro dispositivo, cuyo nombre está compuesto por el nombre del dispositivo del disco y un número que identifica la partición. El número va desde 1 a 4 para particiones primarias y extendidas y desde 5 en adelante para particiones lógicas. Ejemplo: dado el siguiente esquema de particionamiento

```
disco SATA 120GB (/dev/sdb)
 |-> Primera Partición primaria Windows 60GB
 |-> Partición extendida Linux 30GB


     |    |-> Primera Partición lógica Linux 15GB
     |    |-> Segunda Partición lógica Linux 15GB


 `-> Segunda Partición primaria Linux 30GB
```

Los dispositivos pueden ser los siguientes:

Dispositivo	Tipo	Tamaño (GB)	Descripción
/dev/sda1	Primaria Windows	60	1ra Partición Primaria
/dev/sda2	Extendida Linux	30	Partición Extendida
/dev/sda5	Lógica Linux	15	1ra Partición Lógica
/dev/sda6	Lógica Linux	15	2da Partición Lógica
/dev/sda3	Primaria Linux	30	2da Partición Primaria

Tabla 3: Nombre de dispositivos para particiones

3.5 . Sistemas de archivos para Linux

EXT2 (Second Extended File System)

Sistema de archivos para el kernel de Linux. Fue el sistema de archivos por defecto de las distribuciones Red Hat Linux, Fedora Core y Debian hasta ser reemplazado recientemente por EXT3.

Tiene un tipo de tabla FAT de tamaño fijo, donde se almacenan los i-nodos. Los i-nodos son una versión muy mejorada de FAT, donde un puntero i-nodo almacena información del archivo (ruta o *path*, tamaño, ubicación física). En cuanto a la ubicación, es una referencia a un sector del disco donde están todos y cada una de las referencias a los bloques del archivo fragmentado. Estos bloques son de tamaño especificable cuando se crea el sistema de archivos, desde los 512B hasta los 4KB, lo cual asegura un buen aprovechamiento del espacio libre con archivos pequeños. Los límites son un máximo de 2TB de archivo, y de 4TB de partición.

EXT3 (Third Extended File System)

Es el más usado en las distribuciones Linux. Puede ser montado y usado como EXT2. Aunque su velocidad y escalabilidad son menores que las de sus competidores como JFS, XFS o ReiserFS, tiene la gran ventaja de permitir actualizar desde EXT2 sin perder los datos almacenados ni tener que formatear el disco y tiene menor consumo de CPU. Este sistema de archivos agrega a EXT2:

- Journaling.
- Índices en árbol para directorios que ocupan múltiples bloques.
- Crecimiento en línea.

EXT4 (Fourth Extended File System)

Las principales mejoras respecto a EXT3 son

- Soporte de volúmenes de hasta 1024PB.
- Soporte añadido de extent.
- Menor uso de CPU.
- Mejoras en la velocidad de lectura/escritura.

XFS

De 64-bit de alto rendimiento creado por SGI (antiguamente Silicon Graphics Inc.) de licencia GNU GPL. Soporta un sistema de archivos de hasta 9 ExaBytes, dependiendo de los límites impuestos por el SO. Es posible aumentar la capacidad de XFS con xfsgrwfs, ideal para particiones LVM (Logical Volume Manager).

Los sistemas de archivos XFS están particionados internamente en *grupos de asignación*, que son regiones lineares de igual tamaño dentro del sistema de archivos. Los archivos y los directorios pueden crear grupos de asignación. Cada grupo gestiona sus i-nodos y su espacio libre de forma independiente, proporcionando escalabilidad y paralelismo —múltiples hilos pueden realizar operaciones de E/S simultáneamente en el mismo sistema de archivos.

JFS (Journaling File System)

De 64-bit con respaldo de transacciones creado por IBM. Está disponible bajo la licencia GNU GPL. Existen versiones para AIX, eComStation, OS/2, Linux y HP-UX.

Utiliza un método interesante para organizar los bloques vacíos, estructurándolos en un árbol y usa una técnica especial para agrupar bloques lógicos vacíos. Al ser un sistema de archivos de 64 bits soporta archivos grandes y particiones LFS (del inglés Large File Support), lo cual es una ventaja más para los entornos de servidor.

ReiserFS

De propósito general, diseñado e implementado por un equipo de la empresa Namesys, liderado por Hans Reiser. Es el sistema de archivos por defecto en algunas distribuciones SuSE, Xandros, Knoppix y otras.

Con la excepción de actualizaciones de seguridad y parches críticos, Namesys ha cesado el desarrollo de ReiserFS (también llamado reiser3) para centrarse en Reiser4, sucesor de este sistema de archivos.

Principales ventajas:

- Reparticionamiento con el sistema de archivos montado y desmontado. Podemos aumentar el tamaño del sistema de archivos mientras lo tenemos montado y desmontado (online y offline).
- *Tail packing*, un esquema para reducir la fragmentación interna.
- Comparado con EXT2 y EXT3 en el uso de archivos menores de 4k, ReiserFS es normalmente más rápido en un factor de 10-15. Esto proporciona una elevada ganancia en las news, como por ejemplo Usenet, caches para servicios HTTP, agentes de correo y otras aplicaciones en las que el tiempo de acceso a archivos pequeños debe ser lo más rápida posible.

Principales desventajas:

- Los usuarios que usen como sistema de archivos EXT2 deben formatear sus discos, aunque no así los que usen EXT3.
- ReiserFS en versiones del kernel anteriores a la 2.4.10 se considera inestable y no se recomienda su uso, especialmente en conjunción con NFS.
- Algunas operaciones sobre archivos (por ejemplo unlink(2)) no son síncronas bajo ReiserFS, lo que pueden causar comportamientos extraños en aplicaciones fuertemente basadas en locks de archivos.
- No se conoce una forma de desfragmentar un sistema de archivos ReiserFS, aparte de un volcado completo y su restauración.

Sistemas de archivos Journaling y no Journaling

Los antiguos sistemas de archivos debían ser escaneados por completo cada vez que ocurría una caída del sistema. Ejemplos de este tipo de sistema de archivos son EXT2 y FAT32.

Los nuevos filesystems siguen el principio de *metadata only*. En vez de una completa comprobación sólo se tienen en cuenta las modificaciones en los metadatos provocadas por las actividades del sistema. Esto ahorra una gran cantidad de tiempo en la fase de recuperación del sistema tras una caída. Las actividades simultáneas que requieren más entradas de protocolo se pueden unir en un grupo, en el que la pérdida de rendimiento del sistema de archivos se reduce en gran medida mediante múltiples procesos de escritura. En esta categoría entran EXT3, EXT4, XFS, JFS, ReiserFS y NTFS.

3.6 . Particionamiento de discos

3.6.1 . Partición boot

***Punto de montaje**:* **/boot/**
***Descripción**:* aquí se encuentran archivos del sistema importantísimos, como los diferentes kernel que tengamos disponibles y los archivos necesarios para que el gestor de arranque le permita levantarse a Linux. Se utiliza una partición separada de la raíz para brindar una mayor seguridad a los datos. Lo que es más, podemos no montar esta partición para asegurar su integridad: la partición sólo se usaría por el gestor de arranque por un breve instante para leer la información necesaria para levantar el sistema, pero nunca montada.
***Sistema de archivos recomendado**:* EXT2 para tener una gran compatibilidad.
Tamaño recomendado: como son todos archivo pequeños, tal vez con 32 ó 64MB (exagerando) sea sufuciente. De todos modos, para estar seguros podemis asignarle 100MB.

3.6.2 . Partición raíz

***Punto de montaje**: /*
***Descripción**:* el directorio raíz de esta partición es el directorio raíz de todo el sistema. El resto de los dispositivos serán montados en subdirectorios de esta partición.
***Sistema de archivos recomendado**:* EXT3 por compatibilidad y relativa eficiencia. Queda a criterio del administrador.
Tamaño recomendado: como en esta partición estarán todo el sistema, 6 ó 7GB debería ser suficiente para que funcione correctamente.

3.6.3 . Partición swap

***Punto de montaje**:* **NINGUNO**
***Descripción**:* en los sistemas Windows existe un archivo (generalmente en el directorio raíz de cada partición) llamado *page.sys (*"*archivo de paginación*"*),* de tamaño variable pero con un máximo determinado por el usuario, que sirve como espacio de intercambio para cuando el sistema operativo se queda sin memoria RAM. Habitualmente, se llama Memoria Virtual al total de memoria RAM más el espacio de intercambio disponibles para que el sistema opere.

Un espacio de intercambio sólo es un trozo de memoria donde se almacenan datos en forma temporal, como variables y otros muchos posibles datos. Se utilizan módulos de memoria RAM porque físicamente es, por lejos, mucho más rápido acceder y escribir datos en ellos que en dispositivos de disco como IDE, SATA, SCSI, SAS, etc. y en este caso lo que se necesita es rapidez. Los datos escritos en espacio de intercambio son binarios.

¿Por qué entonces no se usa RAM en vez de discos rígidos? Por muchas fuertes razones: los datos se pierden junto con la alimentación, la relación volumen de datos-volumen físico es notablemente superior en los discos rígidos y, finalmente, la relación costo económico-volumen de datos.

En Linux también podemos especificar un "*archivo de paginación*" o, propiamente hablando, un archivo swap ("*swap file*" o "*archivo de intercambio*"*)* del tamaño que queramos y activarlo en forma permanente para que el sistema operativo haga uso de él cuando lo necesite, pero

sabemos que cada vez que se escriben datos en un sistema de archivos se debe pasar por la lógica del mismo, algoritmos y más algoritmos de control, sin contar con posibles errores, etc. Es por eso que en Linux se usa con más frecuencia esta partición (la partición swap) y no un archivo (aunque se pueden usar ambos). Es una partición que no se monta, que no tiene sistema de archivos (no pasa por controles ni lógica) y por lo tanto se escribe en ella con mayor velocidad con excelentes resultados finales.

Sistema de archivos recomendado: **NINGUNO**. Esta partición no debe estar formateada. En los instaladores de algunas distribuciones, cuando estamos particionando en un entorno visual, tal vez nos pregunte de qué tipo, qué formato es el que le queremos dar a nuestra partición, y una opción posible es *swap*. Debe entenderse que

swap no es un formato

Sólo aparece como tal para poder de alguna forma especificar que no debe tener un tipo, que será una partición swap.

Tamaño recomendado: como fórmula general se suele asignarle el doble de la memoria RAM del equipo para estaciones de trabajo y escritorios. Esta fórmula se flexibiliza cuando hablamos de servidores con 4, 8GB ó más de RAM. En estos casos se puede asignar 1 ó 2GB pues es casi ridículo hablar de tanta memoria de intercambio que tal vez no se pueda llegar a direccionar. Casi con seguridad se puede decir que esa swap ni siquiera se va a tocar, salvo tareas extraordinarias que el administrador vaya a hacer.

3.6.4 . Otras particiones

Teóricamente, Linux puede trabajar completamente sobre una sóla partición, montada como la raíz. En este caso, el contenido de **/boot/** está dentro de la misma partición y tendremos que crear un archivo de intercambio, aunque de esta forma estaríamos deshabilitando las grandes ventajas que ya mencionamos.

Así también podemos dar más seguridad a otro directorio que queramos y consideremos importante para la instalación que estamos haciendo. Un ejemplo es el directorio **/home/**, donde se guardan los documentos propios de cada usuario (una descripción completa de directorios importantes en “4.7 . Directorios especiales”).

Otra ventaja que podemos mencionar sobre el montaje es que podemos montar una partición formateada con un sistema de archivos que administra bien archivos pequeños, por ejemplo, en el directorio donde se guardan mails en un servidor de correos, la partición raíz podría estar formateada con un sistema de archivos que maneja bien archivos de tamaño regular, etc. De esta forma el administrador tiene un control más amplio y precisión quirúrgica sobre el sistema completo.

También se puede asegurar particiones de mucha importancia y montarlas como sólo lectura, o montarlas para que sólo algunos usuarios puedan acceder a ella y otros no. Las posibilidades son tantas que escapan al propósito que nos incumbe.

Hay que resaltar una cosa: estando montadas todas las particiones que queramos, nadie (ningún usuario ni el sistema operativo en sí) diferencia los directorios comunes de los

directorios que están sirviendo como puntos de montaje a la hora de recorrer, acceder y escribir archivos (y directorios). Por supuesto, existen comandos para saber qué dispositivos están montados, dónde y en qué forma.

3.6.5 . Compartir el disco con otro SO

Supondremos que compartimos el disco con Windows. Tenemos tres posibles casos:

I. Windows ya está instalado y debe convivir con Linux en el mismo espacio físico (partición) en el que actualmente está instalado. Esto quiere decir que debemos separar la partición de Windows de alguna forma, para que se convierta en al menos dos particiones donde puedan existir los SSOO. Para esto se puede usar una herramienta llamada FIPS que viene con la mayoría de las distribuciones.

FIPS se encuentra en el directorio **/dosutils/** del CD de Linux. Su uso es muy sencillo pero se recomienda tener la precaución de un back up de los archivos más importantes. Antes de correr FIPS sobre la partición, habrá que desfragmentarla y hacer un escaneo de la misma (scandisk). FIPS corre sobre DOS nativo, lo que implica que no puede ser ejecutado desde una tarea DOS lanzada desde Windows.

Aparecerán los datos de las particiones, y luego se presentan dos valores, el primero corresponde a la partición FAT que queremos reducir y, el segundo, a la nueva partición. Con las teclas de cursor derecha e izquierda se cambian los valores. Después de correr FIPS, con *fdisk* de DOS u otra aplicación borramos la partición nueva y ya estamos listos para usarla para Linux.

II. Windows ya está instalado y se dispone de espacio libre para las particiones Linux. En este caso sólo debemos proceder con el particionamiento.

III. Linux es instalado primero y luego se instalará Windows. En este caso debemos tener la precaución de dejar espacio para la partición Windows (tal vez 6 ó 7GB sean más que suficientes en general —queda a conciencia del administrador). Una vez instalado Linux y luego Windows en la partición que le indicamos, deberemos correr nuevamente el comando *grub* desde Linux (ver “7 . Gestor de arranque Grub”).

3.6.6 . Esquema de particionamiento

Antes de comenzar a particionar el disco es recomendable hacer un esquema en papel y lápiz de cómo queremos disponer las particiones. Esto se llama *esquema de particionamiento*, y puede parecerse a la *Tabla 3: Nombre de dispositivos para particiones*, con algunas modificaciones.

Supongamos que: tenemos un disco SATA de 160GB, 1GB de RAM; instalaremos Linux y después Windows; queremos una partición boot, una swap, obviamente necesitamos una partición raíz y, además, crearemos una partición para /home/ para guardar nuestros documentos en una partición independiente.

A la partición Windows, que sólo queremos que tenga el SO y los programas, le asignaremos 15GB, al igual que a la partición raíz de Linux. A la partición boot le damos 100MB, a la partición swap 2GB y a la partición /home/ (que la compartirán Windows y Linux) le damos el espacio restante. Una recomendación personal que se puede hacer es que la primera partición

siempre sea la partición Windows.

Un esquema de particionamiento que sigue las anteriores directrices puede ser:

Dispositivo	Tipo	Tamaño	Sistema de archivos	Punto de montaje	Descripción
/dev/sda1	Primaria	15GB	NTFS o FAT	/mnt/win/	Windows
/dev/sda2	Primaria	15GB	EXT3	/	raíz
/dev/sda3	Extendida	2,09GB	NINGUNO	NINGUNO	extendida
/dev/sda5	Lógica	2GB	NINGUNO	NINGUNO	swap
/dev/sda6	Lógica	100MB	EXT2	/boot/	boot
/dev/sda4	Primaria	127,91GB	FAT o EXT2	/home/	documentos

Tabla 4: Ejemplo de esquema de particionamiento

Opcional: montar la partición Windows en Linux para poder acceder a los archivos en esa partición.

Advertencia: Linux tiene total soporte de lectura/escritura sobre FAT, pero tiene total soporte de lectura y soporte parcial (experimental) de escritura sobre NTFS. Esto quiere decir que si la partición Windows es FAT Linux no tendrá problema en manipularla, pero si es NTFS sólo es seguro poder leerla.

Opcional: la partición de documentos puede estar formateada como FAT y ambos sistemas operativos tendrán acceso total a ella, pero es sabido que EXT2 es probadamente superior a FAT. El problema es que Windows no tiene soporte directo sobre EXT2, pero hay una solución: los llamados IFS (*Installed File Systems*, o *Sistemas de Archivos Instalados*). Se puede instalar un driver para Windows para manipular sistemas de archivos EXT2 con toda seguridad. Para más información visite **http://www.fs-driver.org/**, desde donde puede descargar el último driver.

3.7. Instalación de la distribución seleccionada

Durante la instalación visual de la distribución de nuestra elección se nos hará diferentes preguntas relacionadas. Algunas de ellas pueden ser:

- **Licencia (Opcional)**: en algunas distribuciones hay que aceptar un contrato de licencia durante la instalación, que puede especificar los alcances de la distribución y el soporte y otras especificaciones legales importantes.

Es muy importante que leamos estas condiciones, aunque muchos simplemente la aceptan sin leerlas.

- **Idioma**: nuestro idioma. Se utilizará para ofrecernos un sistema tan traducido a nuestro lenguaje como sea posible (obviamente los comandos no son traducidos).
- **Distribución del teclado**: cómo es nuestro teclado: Inglés, Internacional, Español, etc.
- **Zona horaria**: el huso horario en el que nos encontramos. Argentina es GMT-3 (Greenwich Mean Time menos tres horas) para la *hora del Oeste*. Se puede especificar, por ejemplo, *America/Argentina/Tucuman*.

- **Particionamiento**: se nos ofrecerá un particionador gráfico de discos. La forma en que éste trabaje varía dependiendo de la distribución y de la versión.
- **Usuarios**: tal vez podamos especificar uno o más usuarios, sus contraseñas, etc. Como mínimo, tenemos que tener la posibilidad de establecer la contraseña del usuario *root*.

4 . Entorno básico del shell de Linux

4.1 . Login

Antes de comenzar a trabajar con Linux debemos primero loguearnos con un usuario previamente acreditado. El usuario que persiste en todos los sistemas Linux es el usuario **root**. Este usuario (conocido como superusuario) tiene acceso a absolutamente todo en el sistema, y si algo puede ser hecho por un usuario de seguro que este usuario puede hacerlo. Si nosotros somos los administradores del sistema entonces nos podremos loguear como root.

En pantalla veremos algo como

```
Linux 2.6.24-gentoo-r8 (diego0) (tty1)
diego0 login:
```

Primero aparece la versión del kernel de Linux y la distribución, seguidos del nombre de host y de la terminal en la que se está por loguear. En una nueva línea está el nombre de host nuevamente seguido de " login: ", esperando que escribamos el nombre del usuario con el que nos vamos a loguear. Si es root, lo tipeamos y presionamos la tecla <Entrar>. Luego escribimos la clave (o contraseña) del usuario, presionamos <Entrar> nuevamente y, si hicimos todo bien, se verá algo como lo siguiente

```
Linux 2.6.24-gentoo-r8 (diego0) (tty1)
diego0 login: root
Password:
Last login: Thu Feb 19 10:59:10 on tty2
diego0 ~ # _
```

El encargado del logueo es un programa llamado genéricamente *getty*, aunque según la distribución puede ser *agetty*, *mingetty* u otro.

4.2 . Prompt

A continuación vamos a analizar el prompt que nos devuelve el shell de Linux una vez logueados.

Prompt es la etiqueta que nos indica que el shell está listo para recibir órdenes.

Shell o CLI ("Command Line Interface"), también conocido como intérprete de órdenes, intérprete de línea de órdenes, intérprete de comandos, terminal o consola es, genéricamente, un programa informático que actúa como interfaz

de usuario para comunicarle al sistema operativo órdenes escritas por teclado.

Tanto el prompt de Linux como el popular shell que vamos a usar (llamado *bash* por "Bourne Again SHell") es completamente personalizable y puede variar de distribución en distribución y de acuerdo a los cambios a los que lo sometamos. Bash también es un potente entorno de programación de scripts, y nos permite automatizar muchas tareas.

El siguiente promt de ejemplo muestra:

- El nombre del equipo.
- Un símbolo arroba (@).
- El usuario logueado.
- Un espacio y la carpeta actual en la que se está. Si se está en el directorio personal del usuario logueado, se verá un símbolo "uñuflo" (~), también mal conocido como "tilde" (**tilde es el punto sobre la "i latina"**).
- Un espacio y un símbolo numeral (#) si es que se está logueado como root, o un símbolo peso ($) si se está logueado como otro usuario. Puede ser que cuando estemos logueados como root no se escriba la arroba y el usuario (es decir, no se escribe @root).
- Un espacio y el cursor, que puede ser un guión bajo o un rectángulo (puede estar parpadeando). Indica el punto de inserción actual de caracter, es decir, dónde se escribirá el próximo caracter que ingresemos.

```
diego0@diego $ _
```

4.3 . Cambiar entre TTY's

TTY es un tipo de terminal en los sistemas UNIX, Linux y otros sistemas.

En Linux, generalmente, disponemos de doce tty's accesibles con las teclas CTRL + ALT + [F1 ... F12], de las cuales (con seguridad) las primeras seis (CTRL + ALT + [F1 ... F6]) funcionan en modo texto. Por defecto, encontraremos que la séptima es la primera tty gráfica.

Pero todo esto es altamente configurable y variable.

4.4 . ¿Qué es un comando?

Genéricamente, un comando (del inglés "command", orden, instrucción) es una instrucción o mandato que el usuario proporciona a un sistema informático desde línea de comando (como un shell) o desde una llamada de programación. Puede ser interno (propio del intérprete) o externo (contenido en algún archivo

de tipo ejecutable, script, etc.). Suele admitir parámetros de entrada (argumentos) para modificar el comportamiento predeterminado.

En los sistemas Linux los argumentos que son opciones se indican con un prefijo '-' y '--', respectivamente para argumentos cortos y largos (un argumento corto puede ser 'v', y uno largo, 'version', y ambos pueden tener el mismo efecto final sobre el comando). Ésta es una convención muy popular, aunque en algunos comandos estos argumentos pueden o deben escribirse sin guiones regulares precediéndolos.

En el prompt se escriben comandos.

Para pasar una o más opciones cortas se escribe el comando, un guión regular y la lista de opciones cortas. Para pasar una opción larga se escribe dos guiones regulares seguidos y la opción. Si una opción en sí requiere un parámetro extra se lo escribe por separado de las demás opcioness (si es que es corto, y precedido de un guión) e inmediatamente después el parámetro que necesita. Ejemplo:

```
diego0@diego $ comando -aseg --Arg-Largo -d /dev/sda
--config-file /etc/comando/comando.conf
```

4.5 . Entrada, Salida y Error Estándar

Cuando ejecutamos un comando en Linux se abren tres archivos: Entrada Estándar (stdin), Salida Estándar (stdout) y Error Estándar (stderr). Los programadores de C y C++ se sentirán familiarizados.

La Entrada Estándar son los argumentos que recibe un comando cuando es ejecutado. Puede tratarse de archivos y/o argumentos (que no sean archivos) pasados al comando cuando se lo ejecutó. También puede tratarse de algún dispositivo de entrada, etc.

La Salida Estándar es la salida por defecto, es decir, lo que produce como resultado. Puede ser escrito en pantalla, en un archivo, etc. (o en varios lugares simultáneamente).

El Error Estándar son los errores que reporta un programa. Tiene un tratamiento similar a la Salida Estándar.

4.6 . Comandos básicos

4.6.1 . Camino absoluto y camino relativo

Cuando especificamos la ruta a un archivo o carpeta lo podemos hacer de dos formas: absoluta o relativamente.

Si lo hacemos absolutamente, especificaremos cómo llegar a ese archivo o carpeta desde el

directorio raíz. Es decir, si hablamos del archivo *hola.txt* ubicado en el directorio *diego/*, que, a su vez, se encuentra en el directorio *home/*, contenido en el directorio raíz, entonces el camino absoluto del archivo hola.txt es **/home/diego/hola.txt**.

Si lo hacemos relativamente, entonces debemos indicar cómo llegar a él desde la ubicación actual. Si la ubicación actual es **/home/diego2/**, entonces la ubicación relativa del archivo *hola.txt* mencionado con anterioridad es **../diego/hola.txt**. El directorio **../** se refiere, siempre, al directorio que contiene al directorio del que estamos hablando (no necesariamente del actual). Es decir, que en la dirección relativa anterior estamos diciendo que el archivo *hola.txt* se encuentra en la carpeta **diego/**, que se encuentra en la carpeta que contiene a la carpeta actual. Otra aclaración relacionada que debemos hacer es que el directorio **./** es una referencia al mismo directorio en el que se encuentra la actual referencia (no necesariamente el directorio actual). Así, siguiendo el ejemplo anterior, también podríamos referenciar relativamente al archivo *hola.txt* como **./../diego/hola.txt**, **../diego/./hola.txt**, **./../diego/./././hola.txt** y cuantas veces más queramos añadirle **./**.

Dicho más claramente

todos los directorios contienen una referencia a sí mismos, denotada por '.', y una referencia a su directorio padre, denotada por '..'.

Una pregunta que puede surgir: ¿el directorio raíz tiene una referencia a su directorio padre? Si es así, ¿a dónde nos lleva esa referencia? La respuesta es simple: existe una excepción, el directorio raíz sí tiene una referencia '..', pero referencia a sí mismo.

4.6.2 . Algunos metacaracteres

Antes de comenzar de lleno con el bombardeo de comandos, es importante conocer algunos metacaracteres que nos ayudarán mucho:

- *** (asterisco)**: es un *wildcard* (traducido para este caso como “caracter comodín”). Indica que en esa posición puede haber cualquier cantidad de caracteres cualesquiera.
- **? (signo de pregunta cerrado)**: otro *wildcard*. Indica que en esa posición puede haber a lo sumo un caracter cualquiera.
- **[*ascii*] (corchetes)**: al igual que los anteriores es utilizado dentro de patrones de búsqueda. Sirve para indicar exactamente un caracter ASCII que está dentro de la secuencia dada. Por ejemplo, **[abrt19]** se refiere a cualquiera de los caracteres 'a', 'b', 'r', 't', '1' o '9'. Se pueden indicar sucesiones de la forma **[1-9]** o **[d-p]**, siendo reemplazados por cualquier caracter que entre en la serie (incluyendo los extremos). También es posible especificar combinaciones de ambos, como **[adn-p15-8]**.
- **\ (barra invertida)**: sirve para insertar **secuencias de escape**, es decir, que está seguida de un caracter realiza una función particular, pero que se quiere en este caso represente el mismo carácter. Por ejemplo, '\?' quiere decir '?'. Si se escribiera '?' solo entonces estaríamos hablando del *wildcard*.

4.6.3 . Comandos

Comandos para recorrido y listado de directorios

- *pwd*: escribe el nombre completo del directorio en el que se está trabajando (directorio actual). Dicho de otra forma, escribe el camino absoluto del directorio actual.
- *cd*: tiene el poder de cambiar el directorio actual. Se puede usar de varias formas según sus opciones:
  - Sin opciones cambia el directorio actual al directorio personal del usuario que ejecutó la orden.
  - Con un camino absoluto o relativo como argumento cambia el directorio actual al directorio especificado de ser posible.
  - Si como argumento único tiene un guión regular ("-") cambia el directorio actual al directorio anterior e imprime su nombre completo.
- *ls*: lista el contenido de directorios. Opciones posibles:
  - Si entre sus opciones no se encuentra ningún directorio ni archivo entonces listará el contenido del directorio actual.
  - Si entre sus opciones se encuentra algún archivo entonces se lo listará y si se trata de un directorio entonces se listará su contenido. Puede indicarse más de un directorio y/o archivo.
  - *-a* : no ignora las entradas que comienzan con '.'.
  - *--color[=CUÁNDO]*: define si se utilizan colores para distinguir entre tipos de archivos. CUÁNDO puede ser 'never' (nunca), 'always' (siempre) o 'auto'.
  - *-l*: usa un formato de listado detallado.
  - *-h, --human-readable*: junto con la opción *l*, imprime en un formato legible los tamaños (por ejemplo: 1KB, 234M, 3G, etc.).
  - *i, --inode*: imprime el número de índice de cada archivo.

Comandos para manipular la terminal

- *clear*: limpia la pantalla de la terminal.

Comandos para trabajar con directorios y archivos

- *mkdir*: crea un directorio si es que no existe.
- *rmdir*: borra un directorio, si está vacío.
- *rm*: borra archivos o directorios. Opciones:
  - *-f, --force*: fuerza el borrado, ignora los archivos que no existen, no pregunta.
  - *-i*: pregunta antes de cada borrado.
  - *--one-file-system*: cuando se borra una jerarquía recursivamente, saltea cualquier directorio que se encuentre en un sistema de archivos diferente del que se especificó.
  - *--preserve-root*: no borrar /.
  - *--no-preserve-root*: no tratar en forma especial /.
  - *-r, -R, --recursive*: borrar directorios y su contenido recursivamente.
  - *-v, --verbose*: modo verboso.
- *file*: determina el tipo de archivo de un archivo.
- *cat*: concatena archivos y los escribe en la salida estándar. Opciones
  - *-n, --number*: enumera todas las líneas de la salida.
- *more*: es un filtro para mostrar texto una pantalla a la vez.

- *less*: opuesto de *more*. Similar a éste último, pero con muchas más funcionalidades, como el desplazamiento hacia arriba. Una vez en el programa:
  - */[PATRÓN]*: realiza una búsqueda de PATRÓN.
  - *n*: busca de nuevo.
  - N: busca de nuevo hacia arriba.
  - *q*: termina el programa.
- *head*: escribe en la salida estándar la primera parte (cabeza o "head") de archivos. Por defecto escribe las primeras 10 líneas. Opciones:
  - *-n, --lines=[-]N*: imprime las primeras N líneas en vez de 10. Si a N lo precede -, imprime todo menos las últimas N líneas.
- *tail*: escribe en la salida estándar la última parte (cola o "tail") de archivos. Por defecto escribe las últimas 10 líneas. Opciones:
  - *-f, --follow[={name|descriptor}]*: escribe los datos agregados a medida que el archivo va creciendo. Las opciones *-f, --follow* y *--follow=descriptor* son equivalentes.
  - *-n, --lines=N*: escribe las últimas N líneas en vez de las 10. Se puede usar +N para escribir comenzando desde la N-ésima línea.

Información sobre tamaño particiones montadas

- *du*: estima el espacio de disco que usan archivos. Opciones:
  - *-c, --total*: produce un total (suma todas las cantidades).
  - *-h, --human-readable*: imprime los tamaños en un formato legible.
  - *-s, --summarize*: muestra solamente un total para cada argumento.
  - *-x, --one-file-system*: saltea directorios en diferentes sistemas de archivos.
- *df*: reporta el uso de disco de los sistemas de archivos. Opciones:
  - *-h, --human-readable*: imprime los tamaños en un formato legible.
  - *-t, --type=TIPO*: sólo lista los sistemas de archivo del tipo TIPO.
  - *-T, --print-type*: imprime el tipo de sistema de archivo.
  - *-x, --exclude-type=TIPO*: excluye los sistemas de archivos de tipo TIPO.
- *mount*: poderosísima herramienta para montaje de sistemas de archivos. Sin opciones imprime los sistemas de archivos que están montados, dónde están montados, el tipo de sistema de archivos y en qué forma está montado (**opciones de montaje**).

TTY's

- *chvt N*: cambia la terminal virtual (tty) visible a la terminal N.
- *tty*: escribe el nombre de archivo de la terminal conectada a la entrada estándar.

Comandos varios

- *echo*: escribe una cadena en la salida estándar. Opciones:
  - *-n*: no escribe la nueva línea al final.
  - *-e*: habilita la interpretación de las secuencias de escape precedidas por la barra invertida ("\").
  - *-E*: deshabilita la interpretación de las secuencias de escape precedidas por la barra invertida ("\").
- *free*: muestra información sobre memoria libre y usada en el sistema. Opciones:

- *-b* | *-k* | *-m*: muestra la memoria en Bytes, KB o MB.
- *-t*: muestra una línea más con los totales.

➢ lspci: lista todos los dispositivos PCI e información sobre los mismos. Algunas distribuciones no traen esta herramienta por defecto. El paquete se llama *pciutils* en algunas distribuciones como Gentoo. Opciones:

- *-v, -vv, -vvv*: niveles de verbosidad.
- *-k*: muestra los drivers del kernel que manejan cada dispositivo y también los módulos del kernel capaces de manejarlos.
- *-t*: muestra un diagrama de árbol todos los buses, bridges y dispositivos y las conexiones entre ellos.

➢ *bc*: potente lenguaje de calculadora de precisión arbitraria.

4.6.4 . Redirección de flujos estándares y tuberías (pipes)

Algo muy importante en Linux es la redirección de los flujos estándares a y desde archivos. Esto es, que un comando reciba como entrada el contenido de un archivo, y que un comando pueda devolver un resultado (salida o error) en el contenido de un archivo (que puede o no existir en el momento de la ejecución).

Para el primer caso el archivo debe existir, y se realiza con el metacaracter símbolo “menor que” (“<”) de la siguiente manera:

```
diego0@diego $ comando -asdf --opcion-larga < archivo
```

Esto es la redirección de la entrada estándar.

Para el segundo caso, el archivo puede o no existir (de no existir, es creado). Se puede redireccionar la Salida Estándar y el error estándar, y se puede abrir el archivo en modo escritura (“write”, si existe el archivo, sobreescribirá el contenido del mismo) o en modo agregado (“append”, si existe el archivo, agregará al final del mismo). El metacaracter para escritura es el símbolo “mayor que” (“>”) y el de agregado es dos veces seguidas el símbolo “mayor que”. Para redirigir la Salida Estándar se puede escribir o no el número 1 antes del metacaracter, y para redirigir el Error Estándar se debe escribir el número 2 antes del metacaracter. Ejemplo

```
diego0@diego $ comando -t --opc-larga 2>> /var/log/comando.err.log
```

Tal vez ahora necesitemos que la salida de un comando sea la entrada de otro. Para esto tendríamos que redireccionar la salida del primer comando a un archivo y luego redireccionar la entrada del segundo comando al archivo.

Por suerte existe una forma más directa y eficiente de hacer esto: las tuberías (o “pipes”). Una tubería permite redireccionar la salida de un comando directamente a la entrada de otro. Su sintaxis es

```
diego0@diego $ comando1 | comando2
```

El símbolo de en medio es el llamado *pipe*, y puede resultar conocido para programadores de C y C++ como el operador OR de bits.

4.6.5 . ¿Cómo pedir ayuda?

Es casi seguro que cualquier comando tiene algún tipo de ayuda, y casi siempre es accesible con el parámetro *--help*.

```
diego0@diego $ comando --help
```

También se puede conseguir ayuda de muchos comandos con el comando *man* y con el comando *info*. Se utilizan de la misma forma los dos: se escribe info o man y luego el comando del que queremos ayuda. El hecho de que la descripción de un comando esté más o menos detallada (o no exista) en alguno de los comandos anteriores tiene que ver con la disposición de los desarrolladores del comando del que se quiere ayuda, y por eso se recomienda consultar ambos comandos para mayor (o mejor) información.

4.7 . Directorios especiales

A continuación, una lista de directorios importantes que se podrán encontrar en la mayoría de los sistemas Linux.

- **/bin/** Se guardan los archivos ejecutables en líneas de comando más comunes.
- **/boot/** Archivos necesarios para el buteo de la máquina. Se guardan los kernels, el archivo System.map y otros archivos importantes.
- **/boot/grub/** Aquí se guardan los archivos necesarios para que el gestor de arranque GRUB pueda levantar el sistema, como ser *grub.conf*, stage1 y *stage2*.
- **/dev/** Aquí se encuentran todos los dispositivos del sistema.
- **/etc/** Directorio de configuración del sistema. Muchos (si no todos los) archivos de configuración de Linux y sus servicios y aplicaciones son archivos de texto ASCII, permitiendo administrar el sistema con tan sólo un editor de texto. Hay que resaltar que en otros sistemas, como Microsoft Windows, no es tan fácil.
- **/home/** En este directorio se encuentran los directorios personales de todos los usuarios que no son root.
- **/lib/** Depósito de librerías.
- **/opt/** En este directorio suelen instalarse manualmente los programas.
- **/proc/** En este importante directorio se encuentran directorios numerados, que son los números de los procesos activos. En estos directorios los procesos guardan información que necesitan. De este directorio puede decirse que es una forma de ver al kernel del Linux en ejecución como un conjunto de archivos, algunos editables, que permiten tanto la consulta como la parametrización del kernel en caliente solamente con aplicaciones simples que manejen archivos.
- **/root/** Es el directorio personal del usuario root.
- **/sbin/** Contiene ejecutables binarios para la administración del sistema.
- **/tmp/** Aquí se guardan datos temporales.
- **/usr/** Se guardan datos de usuarios para las aplicaciones, como ejecutables, librerías, imágenes, etc.
- **/usr/bin/** Aplicaciones de usuario.
- **/usr/sbin/** Aplicaciones de usuario para la administración del sistema.

- **/usr/share/** archivos compartidos entre aplicaciones.
- **/usr/src/** Aquí se encuentra el código fuente del kernel y el código fuente de otras aplicaciones importantes para el sistema.
- **/var/** En este directorio se almacenan variables. Siempre está en crecimiento.
- **/var/cache/** Directorio de usado como cache por muchos programas.
- **/var/log/** Aquí se almacenan archivos *log*. Está siempre en franco crecimiento.

Este directorio, al igual que el directorio tmp, se recomienda que sean particiones independientes en servidores.

- **/var/spool/** Aquí se guardan las colas, como la cola de impresión.

4.8 . Tipos de archivos

Existen varios tipos de archivos en Linux. Cuando listamos un archivo regular con el comando *ls -lah --color=always*, en la primera columna veremos qué tipo de archivo es. A continuación se detallan ligeramente y, entre paréntesis, cómo se ve con el comando que acabamos de mencionar:

- *Archivo regular o regular file (-)*: puede ser un archivo de texto, una imagen, un ejecutable, etc.
- *Directorio o directory (d)*: así es, en Linux un directorio también es un archivo, y si observan la quinta columna de la salida del comando ls -lah --color=always, en la que se detalla el tamaño del respectivo archivo, verán que ocupa tan sólo 4KB en la mayoría. Éso es lo que ocupa un directorio solo, su contenido es variable e independiente.
- *Enlace simbólico, symbolic link o symlink (l)*: este tema se ve en “4.9 . Inodos, links duros y links simbólicos”.
- *Dispositivo de bloque o block device (b)*: es un dispositivo que tiene bloques de Bytes, como los discos rígidos o pendrives.
- *Dispositivo de caracter o character device (c)*: permite el acceso a dispositivos en forma de secuencias (o flujos) continuas de Bytes. Por ejemplo, una terminal, impresora, etc.
- *Tubería, pipe o fifo (p)*: este tipo de archivo se usa como comunicación entre procesos, acumulando ordenadamente los mensajes.
- *Enchufe o socket (s)*: permiten la comunicación entre procesos, muchas veces a través de la red. Los sockets de dominio son propios de una máquina; se los referencia como objetos de un sistema de archivos y no como puertos de red. Los archivos "socket" sólo pueden ser leídos o escritos por los procesos involucrados en la comunicación, aunque son visibles como entradas de directorio.

4.9 . Inodos, links duros y links simbólicos

Las siguientes son otras formas de encontrar escrito inodo: nodo índice, i-nodo, nodo-i, inode o i-node. En este manual se usará *inodo*.

Un inodo es una estructura de datos propia del sistema de archivos que contiene las caracteríasticas de un archivo de *cualquier tipo*, como los permisos, fechas, ubucación, etc., **exceptuando el nombre**. Cada inodo está identificado por un número entero (único en el

sistema de archivos) que los directorios recogen en pares con un nombre identificativo que permite acceder al archivo.

Cada archivo tiene un único inodo, pero puede ser accedido desde uno o más nombres.

Esta característica permite más accesibilidad a los archivos dentro del sistema de archivos, y cada nombre que tiene un archivo se llama *enlace duro* ("hard link") o simplemente *enlace* ("link"). Si modificamos un archivo accediéndolo desde uno de sus nombres, cuando lo accedamos desde otro podremos ver los cambios realizados.

Puede ser que un inodo no tenga enlaces. Ese archivo es eliminado y sus recursos liberados (proceso de supresión de archivo), pero si un proceso hubiera estado accediéndolo entonces sólo será eliminado definitivamente cuando la última referencia a él quede cerrada.

Por este motivo, al actualizar un programa, se recomienda primero suprimir el viejo ejecutable y crear a continuación un nuevo archivo e inodo para la versión nueva. La razón es que si la aplicación estaba en ejecución en el instante de la actualización, podrá seguir ejecutándose sin tener que terminarse (leyendo el viejo archivo ejecutable). En el caso en que al archivo ejecutable directamente se le sobreescriba el contenido del nuevo, tendrá que detenerse para leerlo nuevamente, porque se estará trabajando sobre el mismo inodo.

En forma tradicional, era posible crear enlaces a directorios, provocando que la estructura de directorios fuera un **grafo dirigido conexo (con o sin bucles)** en vez de un **árbol (grafo dirigido conexo sin bucles)**. Se podía dar la paradoja de que un directorio fuera su propio padre (o su propio hijo...). Los sistemas modernos generalmente prohíben estas confusiones.

Los enlaces duros de un archivo están todos siempre en el mismo sistema de archivos, pues sólo son una forma de acceder a un archivo a través de su inodo, y tanto los inodos como los archivos están siempre en un mismo sistema de archivos.

Existe otro tipo de enlace llamado *enlace simbólico*, *symbolic link* o *symlink*, que no necesita estar en un mismo sistema de archivos, pueden ser enlaces simbólicos a directorios (a cualquiera), pero nada tienen que ver con el archivo al que referencian. De hecho, no son más que eso: una referencia, un archivo *A* que referencia a un archivo *B*. Cuando editemos un enlace simbólico a archivo, seguramente editaremos el archivo al que referencia.

Para los que estén familiarizados con el sistema operativo de *Redmond* (Windows), un enlace simbólico es muy similiar a un *acceso directo*, pero con algunas características importantes.

Un enlace simbólico es interpretado y tratado como el mismo archivo que referencia tanto por usuarios como por programas, a menos que se pongan en marcha mecanismos para detectar enlaces simbólicos. Por ejemplo, si se elimina un enlace simbólico no afecta en nada al archivo referenciado.

Por otro lado, si se elimina el enlace (duro) del archivo referenciado, el enlace simbólico

estará roto (y tal vez pueda causar algún problema) ya que este enlace sólo referencia a cualquier archivo que cumpla con la condición de tener un enlace duro con el nombre que está referenciando, sin importar qué inodo tenga.

Por esta versatilidad y por la portabilidad de un sistema de archivos a otro rara vez se usan enlaces duros y, casi siempre, enlaces simbólicos.

A continuación, pueden explicarse los siguientes comandos:

- *ln*: crea enlaces duros y simbólicos. Se puede especificar solamente el *target*, en cuyo caso el enlace se crea en el directorio actual, o se puede escribir a continuación el nombre del enlace. Opciones:
  - *-s*: crea un enlace simbólico en vez de enlace duro (por defecto).
- *cp*: copia archivos y carpetas, pasándolos como una lista, a un destino dado al final de la lista. Opciones:
  - *-b*: crea un backup de cada archivo de destino existente.
  - *-f, --force*: si un archivo de destino no puede ser abierto, borrarlo e intentarlo de nuevo.
  - *-i*: pregunta antes de sobreescribir.
  - *-l, --link*: crea enlaces de los archivos en vez de copiarlos.
  - *-R, -r, --recursive*: copia directorios recursivamente.
  - *-s, --symbolic-link*: crea enlaces simbólicos en vez de copiar.
  - *-t, --target-directory=DIRECTORIO*: copia todos los archivos de origen a DIRECTORIO.
  - *-u, --update*: copia sólo cuando el archivo de origen es más nuevo que el archivo de destino, o cuando no existe el archivo de destino.
  - *-v, --verbose*: modo verboso.
  - *-x, --one-file-system*: mantenerse en el mismo sistema de archivos.
- *mv*: mueve un lista de archivos a un directorio especificado al final de la lista. Si tanto el origen como el destino están en un mismo sistema de archivos, sólo tiene que modificar el nombre de archivo (el enlace), por eso este comando también se usa para renombrar archivos.
  - *-b*: hace un backup de cada archivo de destino.
  - *-f, --force*: no pregunta antes de sobreescribir.
  - *-i, --interactive*: pregunta antes de sobreescribir.
  - *-u, --update*: mueve sólo cuando el archivo de origen es más nuevo que el archivo de destino, o cuando no existe el archivo de destino.
  - *-v, --verbose*: modo verboso.

4.10 . Sistema de permisos

A diferencia de DOS, Linux fue creado para trabajar en un entorno multiusuario. Muchas versiones de Windows heredan el sistema de permisos de DOS, y Linux hereda de UNIX.

En Linux existen usuarios y grupos de usuarios, en donde se agrupan usuarios. Un usuario debe pertenecer a un grupo de usuarios, llamado grupo de logueo, y, opcionalmente, puede pertenecer a cualquier otro grupo. Cada grupo y cada usuario tienen asignado por aparte un número único que los identifica (ID), una contraseña (para el usuario) y al menos un nombre que referencia al usuario. Un usuario puede tener dos o más nombres, es decir, hay dos o más formas de acceder al mismo usuario.

Un archivo tiene un propietario ("owner") y un grupo ("group") propietario. Los permisos se establecen respecto al propietario, el grupo propietario y los "otros" ("others"), que son los usuarios que no son el propietario ni pertenecen al grupo propietario. Este sistema de permisos se llama **UGO (User, Group, Other)**.

Los siguientes permisos se pueden establecer para cualquiera de ellos:

- *Read (r)*: permiso de lectura.
- Write (w): permiso de escritura.
- Execute (x): permiso de ejecución.

Cuando ejecutamos el comando *ls -lah --color=always* podremos ver en las nueve columnas que suceden a la columna de tipo de archivo (la primera) los permisos del archivo en ternas ordenadas respetando el orden UGO. Cada elemento de la terna puede ser la correspondiente letra si es que ese permiso está activado, o un guión regular si es que no lo está. La siguiente columna indica la cantidad de enlaces que tiene ese archivo, la siguiente, el propietario, y la siguiente el grupo propietario.

En las tres columnas de permiso de ejecución puede haber otra letra, indicando permisos especiales. Éstos son:

- SUID (Set User ID) (s): permiso para que cualquier usuario ejecute el archivo como si fuera el propietario. **¡Cuidado con este permiso!**
- SGID (Set Group ID) (s): permiso para que cualquier usuario ejecute el archivo como si perteneciera al grupo propietario.
- Sticky Bit o Restricted Deletion Bit (t): la interpretación varía según el tipo de archivo.

Para directorios, previene que un usuario sin privilegios elimine o renombre un archivo en el directorio, a menos que sea el propietario del archivo o del directorio en cuestión; éste es el llamado bandera de deleción restringida ("restricted deletion flag") para el directorio, y es comúnmente encontrada en directorios de acceso para lectura globales como **/tmp/**. Para archivos regulares en algunos sistemas antiguos, el bit guarda la imagen del texto del programa en el dispositivo swap para que pueda cargar con mayor velocidad cuando se ejecute; éste se llama bit pegajoso ("sticky bit").

Cada uno de estos permisos, si están establecidos, se ubican en la columna de ejecución en lugar de la *x* o el guión regular, respectivamente, en el orden UGO.

Considerar el siguiente ejemplo compilatorio:

```
diego0 ~ # cd /
diego0 / # ls -lah --color=always
total 85K
drwxr-xr-x  22 root  root 4,0K mar  3 09:39 .
drwxr-xr-x  22 root  root 4,0K mar  3 09:39 ..
drwxr-xr-x   2 root  root 4,0K feb 16 10:15 bin
drwxr-xr-x   4 root  root 1,0K feb  2 20:19 boot
drwxr-xr-x  14 root  root  14K mar  3 09:13 dev
drwxr-xr-x   5 diego root 4,0K feb 13 14:34 DIEGO
drwxr-xr-x  72 root  root 4,0K mar  3 09:13 etc
drwxr-xr-x   3 diego root 4,0K nov 20 09:26 home
drwxr-xr-x   8 root  root 4,0K oct 22 14:29 lib
drwx------   2 root  root  16K may 15  2008 lost+found
```

```
drwxr-xr-x   3 root  root 4,0K mar  3 09:13 media
drwxr-xr-x   4 root  root 4,0K abr 23  2008 mnt
drwxr-xr-x   9 root  root 4,0K feb 11 09:39 opt
dr-xr-xr-x 104 root  root    0 mar  3 09:12 proc
drwx------  31 root  root 4,0K mar  3 09:26 root
drwxr-xr-x   2 root  root 4,0K oct  9 11:12 sbin
drwxr-xr-x   2 diego root 4,0K mar  3 09:15 sistema
drwxr-xr-x  12 root  root    0 mar  3 09:12 sys
drwxrwxrwt  10 root  root 4,0K mar  3 09:29 tmp
drwxr-xr-x  16 root  root 4,0K jul 15  2008 usr
drwxr-xr-x  13 root  root 4,0K jun 18  2008 var
drwxrw-rw-   3 diego root 4,0K ene 29 12:04 virtual
diego0 dev # cd dev/
diego0 dev # l sda* tty? std*
brw-r----- 1 root disk 8, 0 mar  3 09:13 sda
brw-r----- 1 root disk 8, 1 mar  3 09:13 sda1
brw-r----- 1 root disk 8, 2 mar  3 09:13 sda2
brw-r----- 1 root disk 8, 3 mar  3 09:13 sda3
lrwxrwxrwx 1 root root    4 mar  3 09:13 stderr -> fd/2
lrwxrwxrwx 1 root root    4 mar  3 09:13 stdin -> fd/0
lrwxrwxrwx 1 root root    4 mar  3 09:13 stdout -> fd/1
crw--w---- 1 root root 4, 0 mar  3 09:13 tty0
crw------- 1 root root 4, 1 mar  3 09:13 tty1
crw------- 1 root root 4, 2 mar  3 09:13 tty2
crw------- 1 root root 4, 3 mar  3 09:13 tty3
crw------- 1 root root 4, 4 mar  3 09:13 tty4
crw------- 1 root root 4, 5 mar  3 09:13 tty5
crw------- 1 root root 4, 6 mar  3 09:13 tty6
crw--w---- 1 root root 4, 7 mar  3 09:13 tty7
crw--w---- 1 root tty  4, 8 mar  3 09:13 tty8
crw--w---- 1 root tty  4, 9 mar  3 09:13 tty9
diego0 dev # cd ../bin/
diego0 bin # l [m-p]*
-rwxr-xr-x 1 root root 9,4K jun  4  2008 mbchk
-rwxr-xr-x 1 root root  22K feb 12  2008 mkdir
-rwxr-xr-x 1 root root  18K feb 12  2008 mkfifo
-rwxr-xr-x 1 root root  22K feb 12  2008 mknod
-rwxr-xr-x 1 root root 5,4K jun  6  2008 mktemp
-rwxr-xr-x 1 root root  30K feb 12  2008 more
-rws--x--x 1 root root  79K feb 12  2008 mount
-rwxr-xr-x 1 root root 5,3K feb 12  2008 mountpoint
-rwxr-xr-x 1 root root  62K feb 12  2008 mv
-rwxr-xr-x 1 root root 138K feb 12  2008 nano
-rwxr-xr-x 1 root root  98K may 19  2008 netstat
lrwxrwxrwx 1 root root    8 may 19  2008 nisdomainname -> hostname
-rws--x--x 1 root root  29K feb 12  2008 passwd
lrwxrwxrwx 1 root root   20 may 15  2008 pgawk -> /usr/bin/pgawk-3.1.5
lrwxrwxrwx 1 root root   16 may 15  2008 pidof -> ../sbin/killall5
-rws--x--x 1 root root  30K feb 12  2008 ping
```

```
-rws--x--x 1 root root  26K feb 12  2008 ping6
-r-xr-xr-x 1 root root  66K feb 12  2008 ps
-rwxr-xr-x 1 root root  18K feb 12  2008 pwd
diego0 bin # _
```

Como podrán ver, cuando lista un enlace simbólico escribe la línea con toda la información del archivo y luego imprime " -> " y el nombre del archivo que referencia.

Comandos relacionados

- chgrp: cambia el grupo propietario de los archivos indicados. Opciones:
  - *--no-preserve-root*: no tratar el directorio raíz en forma especial.
  - *--preserve-root*: tratar el directorio raíz en forma especial.
  - -f, --silent, --quiet: suprime la mayoría de los mensajes de error.
  - --reference=ARCHREF: usa el grupo propietario de ARCHREF en vez de especificarlo directamente.
  - -R, --recursive: operar en archivos y directorios recursivamente.
  - -v, --verbose: escribe un diagnóstico para cada archivo procesado.
- chown: cambia el propietario y/o el grupo propietario de archivos. Si sólo un propietario (un nombre de usuario o ID de usuario) es dado, ese usuario será el propietario de cada archivo dado, y el grupo propietario de los archivos no se cambia. Si al propietario lo adyace un símbolo "dos puntos" (":") y el nombre de un grupo (o ID numérico de grupo), sin espacio entre ellos, el grupo propietario de los archivos también es cambiado. Si se escribe el símbolo "dos puntos" pero no el grupo, ese usuario será el propietario de los archivos y el grupo propietario de los mismos será cambiado al grupo de logueo del usuario dado. Si son dados el símbolo "dos puntos" y el grupo y el usuario es omitido, entonces sólo el grupo propietario de los archivos es cambiado; en este caso, chown realiza la misma tarea que chgrp. Si sólo se brinda el símbolo "dos puntos", o si todo el operando está vacío, no se cambia el propietario ni el grupo propietario de los archivos.

  Opciones:
  - *--from=ACTUAL_PROP:ACTUAL_GRUPO*: cambiar el propietario y/o grupo propietario de los archivos sólo si el actual propietario y grupo propietario de ellos encaja con los aquí dados. También puede ser omitido, en cuyo caso no se necesita el patrón para el atributo omitido.
  - *--no-preserve-root*: no tratar el directorio raíz en forma especial.
  - *--preserve-root*: tratar el directorio raíz en forma especial.
  - -f, --silent, --quiet: suprime la mayoría de los mensajes de error.
  - --reference=ARCHREF: usa el propietario y grupo propietario de ARCHREF en vez de especificar los valores PROPIETARIO:GRUPO.
  - -R, --recursive: operar en archivos y directorios recursivamente.
  - -v, --verbose: escribe un diagnóstico para cada archivo procesado.
- chmod: cambia los bits de modo (permisos) de los archivos. Su sintaxis es:

```
diego0 bin # chmod <opciones> <nuevos_permisos> <lista_archivos>
```

Hay dos formas de describir los nuevos permisos: la forma simbólica y la forma octal, que es la que usaremos. En la forma octal los permisos se escriben como un número octal de uno hasta cuatro dígitos, y los dígitos que se omiten (si el número tiene menos de cuatro dígitos) se interpretan como ceros a la izquierda. El primer dígito representa los permisos especiales y los tres subsecuentes, respectivamente, los permisos en el orden UGO.

Para el primer dígito, 2^0 (uno) es el valor del *bit de deleción restringida* o *bit pegajoso*, 2^1 (dos), el de *SGID* y 2^2 (cuatro), el de *SUID*. Se suman los valores respectivos de los permisos que se quieren imponer (cero si es que se quiere **no tenerlo**) y resulta un número entre cero y siete (ambos inclusive), una de las ochos posibilidades.

Para cualquiera de los siguientes dígitos, 2^0 (uno) es el valor del *permiso de ejecución*, 2^1 (dos), el de *escritura* y 2^2 (cuatro), el de *lectura*. Los respectivos dígitos se obtienen de la misma forma que el dígito para permisos especiales.

Este sistema parece ser muy confuso, pero a medida que se adquiere práctica, y, sobre todo, a medida que se observan los permisos de los archivos con el comando *ls -lah --color=always* se vuelve muy natural, especialmente para quienes se familiarizan con el sistema binario.

Opciones:

- *--no-preserve-root*: no tratar el directorio raíz en forma especial.
- *--preserve-root*: tratar el directorio raíz en forma especial.
- -f, --silent, --quiet: suprime la mayoría de los mensajes de error.
- --reference=ARCHREF: usa el propietario y grupo propietario de ARCHREF en vez de especificar los valores PROPIETARIO:GRUPO.
- -R, --recursive: operar en archivos y directorios recursivamente.
- -v, --verbose: escribe un diagnóstico para cada archivo procesado.

4.11 . Editor de textos vim

Es un potentísimo, extenso, extensible y flexible editor de textos en modo texto.

Para ingresar se puede tipear *vi* o *vim*, seguido o no de una lista de archivos a editar. Por defecto, al ingresar al programa se está en *modo comando*, lo que quiere decir que lo que tipeemos no será directamente escrito en el documento, sino que será interpretado como un comando. Existe también el *modo edición* o *modo inserción*, que es activado por un comando, en el cual sí se escribe directamente en el documento, y el *modo reemplazo*. En vim, al igual que en todo Linux, se debe respetar la capitalización de palabras y comandos.

A muchos comandos debe especificárseles sobre qué actuar, el objeto sobre el que se quiere realizar la acción. Por ejemplo, si se quiere borrar, hay que decir qué queremos borrar: una línea, una palabra, hasta el final del documento, todo el documento, etc. También puede especificarse cuántas veces se va a realizar el comando tipeando un número natural antes del comando (por defecto este número es 1). Por ejemplo, si la acción final puede ser “35”, “borrar”, “palabra sin incluir espacio al final”, y se borrarán 35 palabras sin incluir el espacio al final. Se tienen en cuenta los objetos partiendo del objeto actual, es decir, del objeto en el que se está posicionado con el cursor.

Teniendo en cuenta la posición actual, los objetos de movimiento (sobre los que recaen acciones) también pueden ser ejecutados como comandos independientes.

Cambio entre modos

Si estamos en *modo comando* (por defecto), con “i”, “I”, “a”, “A” o “<Insertar>” pasamos a *modo inserción*. Con “R” pasamos *modo reemplazo* a partir del final del cursor. Estando en cualquiera de los dos últimos, con “<Escape>” salimos a *modo comando* y con “<insert>” alternamos entre *modo inserción* y *modo reemplazo*.

La diferencia entre “i”, “I”, “a”, “A” y “<Insertar>” es la siguiente:

- i, <Insertar> el punto de inserción está antes del cursor.
- I el punto de inserción está en el principio de la línea.
- a el punto de inserción está después del cursor.
- A el punto de inserción está al final de la línea.

Objetos de movimiento derecha-izquierda

- h, <Izquierda>, CTRL-H, <BS> caracter anterior dentro de la misma línea.
- l, <Derecha>, <Espacio> caracter siguiente dentro de la misma línea.
- 0 (cero), <Inicio> principio de línea.
- ^ primer caracter no blanco de la línea.
- $, <Fin> fin de línea.

Objetos de movimiento arriba abajo

- k, <Arriba>, CTRL-P línea anterior en la misma columna que la actual (si es posible).
- j, <Abajo>, CTRL-N, CTRL-J línea siguiente en la misma columna que la actual (si es posible).
- _ (guión bajo) esta línea al primer carácter.
- - (guión regular) línea anterior al primer carácter.
- + (más) línea siguiente al primer carácter.
- G ir a principio de línea. Si es precedida por un número natural, intenta llegar a la línea con ese número de línea, sino va a la última línea (si es que ese número supera la cantidad de líneas del documento o si no se especifica el número).

Objetos de movimiento entre palabras

- B, b hasta el principio de la palabra. Mueve hasta ese punto.
- E, e hasta el final de la palabra. Mueve hasta ese punto.
- W, w hasta el principio de la siguiente palabra. Mueve hasta ese punto.

Objetos de movimiento dentro del texto

- ((paréntesis abierto) principio de la oración. Mueve hasta ese punto.
-) (paréntesis cerrado) final de la oración. Mueve hasta ese punto.
- { (llave abierta) principio del párrafo. Mueve hasta ese punto.
- } (paréntesis cerrado) final del párrafo. Mueve hasta ese punto.

Deleción

- <supr>, x elimina caracteres bajo y después del cursor. Hace lo mismo que “dl”.
- X elimina caracteres antes del cursor. Hace lo mismo que “dh”.

- d<obj_movimiento> elimina el texto hasta la posición en que <obj_movimiento> desplaza.
- dd elimina líneas a partir de la actual.
- D elimina los caracteres hasta el final de la línea. Sinónimo de “d$”.

Deleción e inserción

- c<obj_movimiento> elimina hasta donde <obj_movimiento> mueve e inicia el modo de inserción.
- cc elimina líneas e inicia modo inserción.
- C igual que “c$”.
- s elimina caracteres e inicia modo inserción. Sinónimo de “cl”.
- S sinónimo de “cc”.

Copia y movimiento de texto

- y<obj_movimiento> copia texto.
- yy copia líneas.
- Y sinónimo de “yy”.
- p pone el texto copiado después del cursor.
- P pone el texto copiado antes del cursor.

Buscar y reemplar (Estando en modo comando)

- /{patrón}[/]<CR> busca adelante la ocurrencia de {patrón}.
- /<CR> busca adelante el último {patrón}.
- ?{patrón}[?]<CR> busca atrás la ocurrencia de {patrón}.
- ?<CR> busca adelante el último {patrón}.
- n repite el último “/” o “?” realizado.
- N repite el último “/” o “?” realizado pero en sentido opuesto.
- CTRL-C Interrumpe el actual comando de búsqueda.
- :<rango>s/viejo/nuevo/[g][c] reemplaza “viejo” por “nuevo” dentro de <rango> siguiendo las opciones [g], [c] (o ninguna).
  - <rango>
    - Puede no estar presente, en cuyo caso buscará en la línea actual.
    - % busca en todo el archivo.
    - <Inicio-B>,<Fin-B> especifica desde dónde y hasta dónde se buscará (qué líneas). $ significa fin del archivo, por lo que “0,$” es lo mismo que “%”.
  - [g] especifica que la búsqueda se haga globalmente dentro de <rango> y no solamente en la línea actual (por defecto).
  - [c] pide confirmación antes de realizar un reemplazo

Otros comandos:

- :w guarda el documento actual.
- :q sale del documento actual.
- :<w|q>a guarda/sale de todos los ducumentos.
- :<Comando>! fuerza a que se ejecute <Comando>.

- :!<Comando> ejecuta <Comando> en el shell y luego retorna a vi.
- ZZ guarda el ducumento actual y luego sale.

4.12 . Filtros y otros

grep

Imprime líneas que contiene un patrón dado, es decir, que la línea satisface el patrón o que contiene una o más subcadenas que lo satisfacen. Opciones:

- *-A NUM, --after-context=NUM*: imprime las NUM líneas de contexto después de cada línea para cada línea que contiene el patrón.
- *-B NUM, --before-context=NUM*: imprime las NUM líneas de contexto antes de cada línea para cada línea que contiene el patrón.
- *-C NUM, --context=NUM*: imprime NUM líneas de contexto de cada línea para cada línea que contiene el patrón.
- *--colour[=CUÁNDO], --color[=CUÁNDO]*: envolver la cadena que satisface el patrón con el marcador que se encuentra en la *variable de entorno* GREP_COLOR. CUÁNDO puede ser "never", "always" o "auto".
- *-c, --count*: suprimir la salida normal; en cambio, imprimir la cantidad de líneas que contienen el patrón. Con la opción *-v, --invert-match* cuenta las que no lo hacen.
- *-e PATRÓN, --regexp=PATRÓN*: usa PATRÓN como patrón de búsqueda. Útil para proteger patrones que comienzan con un guión regular.
- *-i*: ignora la interpretación de la capitalización tanto en el patrón como en la entrada.
- *-n, --line-number*: antepone el número de línea de cada línea que contiene el patrón dentro de su archivo de entrada.
- *-R, -r, --recursive*: lee todos los archivos en cada directorio.
- *-v, --invert-match*: invierte el sentido de la búsqueda, para seleccionar las líneas que **no contienen** el patrón.

sed

Editor de flujos para filtrar y transformar texto. El primer argumento que recibe es una *expresión regular* encerrada entre comillas. Esta expresión regular es muy similiar a la que escribíamos en vi para buscar y reemplazar. Cabe señalar que las expresiones regulares siguen un estándar y se especializan en cada programa.

line

Lee una línea.

cut

Elimina secciones de cada línea de archivos. Imprime las partes seleccionadas de las líneas de cada archivo de entrada en la Salida Estándar. Opciones:

- -d, --delimiter=DELIM: usar DELIM en vez de <Tab> para la delimitación de campos.
- [-b, --bytes | -c, --characters | -f, --fields]=LISTA: seleccionar, respecticamente, sólo

estos bytes/caracteres/campos especificados en LISTA. En el caso de campos, también imprimir cualquier línea que no contenga caracteres delimitadores, a menos que se establezca la opción -s. Cada lista está compuesta de un rango o de varios rangos separados por comas. La entrada seleccionada se escribe en el mismo orden en que se lee, y se escribe exactamente una vez. Entiéndase por elemento a cada byte, carácter o campo. Cada rango puede ser uno de:

- N: N-ésimo elemento comenzando desde 1.
- N-: desde el N-ésimo elemento hasta el final de la línea.
- N-M: desde N-ésimo hasta el M-ésimo elemento (ambos incluidos).
- -M: desde el primer hasta el M-ésimo elemento.

- -s, --only-delimited: no imprimir líneas que no contengan delimitadores.
- --output-delimiter=CADENA: usar CADENA como el delimitador de salida; por defecto, es el mismo que de entrada.

wc

Imprime el conteo de nueva líneas, palabras y bytes para cada archivo. Opciones:

- -c, --bytes || -m, --chars || -l, --lines || -w, --words: imprime, respectivamente, el conteo de bytes, caracteres, nueva líneas y palabras.
- -L, --max-line-length: imprime la longitud de la línea más larga.

find

Busca archivos en una jerarquía de directorios. Uso:

```
diego0 ~ # find [directorio] [opciones] [tests]
```

Opciones:

- -mount, -xdev: no entrar en otros sistemas de archivos.
- -warn, -nowarn: enciende o apaga las advertencias respecto del uso del comando.

Los argumentos numéricos pueden ser especificados como:

- +n: para un número mayor que n.
- -n: para un número menor que n.
- n: exactamente n.

Tests:

- -empty: el archivo está vacío y es un directorio o archivo regular.
- -executable: archivos que son ejecutables y directorios explorables.
- -false: siempre falso.
- -group gname: el grupo propietario del archivo es “gname” (se permite ID de grupo).
- -iname patrón: la base del nombre del archivo encaja con el patrón de shell “patrón” y no discrimina capitalización (“case insensitive”).
- -name patrón: la base del nombre del archivo encaja con el patrón de shell “patrón” y discrimina capitalización (“case sensitive”).
- -mmin n: los datos del archivo fueron modificados por última vez hace “n” minutos.
- -mtime n: los datos del archivo fueron modificados por última vez hace “n”*24 horas.
- -perm modo: sólo pasan este test los archivos que tienen exactamente los bits de modo como “modo”.

- -perm -modo: los bits de permisos establecidos por "modo" están presentes en el archivo, pero puede que tenga más permisos. Si lo vemos desde el punto de vista octal y si especificamos, por ejemplo, -222, entonces pasará este test cualquier archivo que tenga permiso de escritura para cualquier usuario, sin importar si tiene otro permiso como permiso de ejecución para el propietario, etc. Más flexible es esta forma que la anterior.
- -readable: archivos que pueden leerse.
- -regex patrón: el nombre del archivo satisface la expresión regular "patrón".
- -true: siempre verdadero.
- -type c: el archivo es de tipo "c": b, c, d, p, l, s, igual que con ls -lah --color=always, si es un archivo regular, f.
- -user uname: el propietario del archivo es "uname" (se permite ID de usuario).
- -writable: archivos escribibles.

4.13 . Procesos

Un proceso, también llamado *tarea*, es una ejecución, una instancia o también el contexto de un programa. Este contexto puede constar de más procesos hijos, su estado de ejecución (los valores de los registros del CPU), recursos del sistema, atributos de seguridad, etc. Casi todo lo que se ejecuta en el sistema es un proceso, a excepción del kernel, que es un conjunto de rutinas que residen en memoria a las que los procesos pueden acceder con llamadas al sistema.

A cada proceso que se inicia se le es asignado un número entero positivo que lo identifica durante su ejecución y se llama ID de proceso o PID ("Process ID").

Linux es un SO multitarea y multiusuario, por lo que pueden coexistir muchos procesos, instancias de un mismo programa, iniciados por diferentes o por un mismo usuario y no interferir entre ellos; lo que es más, lo más probable es que cooperen. Cada proceso tiene la "ilusión" de ser el único ejecutándose en el sistema y de tener total acceso al mismo.

Diferencias entre hilos y procesos hijos

Al crear un proceso hijo lo que se hace es copiar la memoria del padre en otra dirección y hacer que estos punteros señalen a la nueva; al crear un hilo lo que se hace es copiar los punteros, de forma que todos los hilos de un mismo proceso comparten exactamente el mismo espacio de direcciones. La sincronización y exclusión mutua del acceso concurrente de los hilos a la memoria del proceso es responsabilidad del programador que los usa.

Instantáneas de procesos

Los siguientes comandos muestran una instantánea de los procesos en el momento en que son ejecutados.

- *ps*: muestra información sobre una selección de procesos activos. Opciones:
  - *-e, -ef, -eF, -ely (sintaxis estándar), ax, axu (sintaxis BSD):* muestra todos los procesos en el sistema.
  - *-ejH, axjf*: imprime un árbol de procesos.
  - *-eLf, axms*: información sobre hilos.
- *pstree*: muestra un árbol de procesos. Opciones:

- *-n*: ordena los procesos con un mismo ancestro por PID en vez de por nombre.
- *-p*: muestra el PID de cada proceso como un número decimal entre paréntesis después de su nombre.
- *-u*: mostrar transiciones de uid. Si el uid de un proceso difiere del de su padre entonces lo muestra entre paréntesis después de su nombre.

4.13.1 . Proceso init

Es el primer proceso que se ejecuta, padre de los demás procesos y, por tanto, tiene PID 1. Lo inicia directamente el kernel y sobrevive a la señal 9 (KILL), irresistible para los demás procesos, y es el responsable de iniciar el sistema en el modo requerido, por lo que los demás procesos pueden iniciarse a través de init o de otro proceso descendiente de él.

4.13.2 . Llamadas del sistema

Una llamada del (o al) sistema ("Systema Call") es el mecanismo por el cual un programa solicita un servicio al SO.

- exec: ejecuta comandos y abre, cierra o copia archivos descriptores.

Si exec se especifica con un comando, reemplazará el shell con el comando sin crear un nuevo proceso. Si se especifican argumentos, éstos serán para el comando. La redirección afecta el entorno de ejecución del shell actual. Un ejemplo de esta llamada del sistema es visible cuando nos logueamos en una tty. El proceso que se está ejecutando primero es getty (o alguna variante), y, después de loguearnos, este proceso usa esta llamada para iniciar el proceso login, que tomará su PID e iniciará el shell.

- *fork*: crea un proceso hijo al duplicar el proceso de llamada. El nuevo proceso, llamado hijo, es un duplicado exacto del proceso que hace la llamada, llamado padre, con algunas excepciones como, por ejemplo, el proceso hijo tiene su propio y único PID que no concuerda con el de ningún grupo de procesos existente.

4.13.3 . Código de estado de los procesos

Los siguientes son los más importantes. Una descripción de todos ellos así como una mejor explicación se encuentran con *man 1 ps*. Se puede encontrar estos y otros códigos de estado de los procesos activos con *ps xa*.

- D: sueño ininterrumpible (generalmente E/S).
- R: ejecutando o susceptible de ser ejecutado (en cola de ejecución).
- S: sueño interrumpible (esperando que se termine un evento).
- T: detenido por una señal de control de trabajo o porque está siendo rastreado.
- X: muerto (jamás debería ser visto).
- Z: zombie (terminado pero se mantiene porque el padre no lo terminó correctamente).

Para formatos BSD:

- s: líder de sesión.
- +: está en el grupo de procesos de ejecución en segundo plano.

4.13.4 . Procesos zombies

Un proceso zombie o "defunct" (muerto) es un proceso que ha completado su ejecución pero

aún tiene una entrada en la tabla de procesos, permitiendo al proceso que le ha creado leer el estado de su salida. Metafóricamente, el proceso hijo ha muerto pero su "alma" aún no ha sido recogida.

Cuando un proceso acaba, toda su memoria y recursos asociados a él se desreferencian, para que puedan ser usados por otros procesos. De todas formas, la entrada del proceso en la tabla de procesos aún permanece. Al padre se le envía una señal SIGCHLD indicando que el proceso ha muerto; el manejador para esta señal será típicamente ejecutar la llamada al sistema wait, que lee el estado de salida y borra al zombie. El ID del proceso zombie y la entrada en la tabla de procesos pueden volver a usarse.

Un proceso zombie no es lo mismo que un proceso huérfano. Los procesos huérfanos no se convierten en procesos zombies, sino que son adoptados por init (ID del proceso = 1), que espera a su hijo.

4.13.5 . Enviar señales

Una señal es una forma limitada de comunicación entre procesos, una notificación asíncrona enviada a un proceso para informarle de un evento. Cuando se le manda una señal a un proceso, el sistema operativo modifica su ejecución normal. Si se había establecido anteriormente un procedimiento (handler) para tratar esa señal se ejecuta éste, si no se estableció nada previamente se ejecuta la acción por defecto para esa señal.

La siguiente tabla muestra las señales estándar soportadas por Linux (más información en *man signal(7)*). Los números de las señales muchas veces dependen de la arquitectura, como indica la columna “valor”: el primer valor es para *alpha* y *sparc*, la segunda, *i386*, *ppc* y *sh*, y la tercera, *mips*.

El valor de la columna acción puede ser cualquiera de:

- Term: terminar el proceso.
- Ign: ignorar la señal.
- Core: terminar el proceso y hacer un volcado del sistema.
- Stop: detener el proceso.
- Cont: terminar el proceso si actualmente está detenido.

Señal	Valor	Acción
SIGHUP	1	Term
SIGINT	2	Term
SIGQUIT	3	Core
SIGABRT	6	Core
SIGKILL	9	Term
SIGSEGV	11	Core
SIGALRM	14	Term
SIGTERM	15	Term
SIGCHLD	0,17,18	Ign
SIGSTOP	17,19,23	Stop

Tabla 5: Señales estándar soportadas por Linux

Envío de señales por el usuario

Es posible enviar señales directamente a los procesos por medio del teclado y por comandos. Las siguientes señales es posible enviarlas a través del teclado:

- SIGINT (2): CTRL + C.
- SIGSTOP (17,19,23): CTRL + Z.

También podemos usar los comandos *kill* y *killall*, de los cuales explicaremos el primero: *kill* envía una señal a un proceso. Recibe como entrada el PID de un proceso (al que se le enviará la señal) y, si no se explicita una señal, envía la señal SIGTERM (15). Para hacerlo, podemos escribir el nombre de la señal (en mayúsculas claro está) o su correspondiente valor antecedido por un guión regular.

Ejemplos de envíos de señales con comandos

Para este experimento se necesita tener sólo dos terminales activas, que pueden (y en este ejemplo será así) ser las tty 1 y 2.

En la tty1 ejecutamos *ps* sin argumentos y debería resultar algo como

```
  PID TTY          TIME CMD
 5012 tty1     00:00:00 login
 5361 tty1     00:00:00 bash
 5368 tty1     00:00:00 ps
```

Ahora sabemos que el PID del proceso bash que se ejecuta en la tty1 es 5361. A continuación, en la misma tty, presionamos varias veces la combinación de teclas CTRL+C (que envía la señal 2, SIGINT) y vemos que bash responde devolviendo el prompt cada vez que enviamos la señal. Luego limpiamos la pantalla con el comando *clear* y nos pasamos a la tty2.

Ahora vamos a enviarle la misma señal al mismo proceso (el proceso bash que se ejecuta en la tty1) pero con el comando kill desde la tty2. Para hacerlo, escribimos kill -2 y el número de proceso (en este caso 5361). Repetimos varias veces, supongamos diez, y nos pasamos a la tty1 y comprobaremos que efectivamente hay diez prompts más, devueltos por bash durante las respectivas veces que le enviamos la señal 2.

Finalmente mataremos el bash de la tty1 *desde la tty1*. Primero lo intentaremos con la señal 15 (SIGTERM) y veremos que bash la resiste. Ahora probemos con la señal 9 (SIGKILL), y veremos que efectivamente bash es terminado.

4.13.6 . Ejecución de programas en segundo plano

Es posible ejecutar tareas en segundo plano agregando el símbolo “ampersand” (“&”) al final del comando. Ésto significa que podemos ejecutar una tarea larga (como la compilación de un programa grande) y que se nos devuelva el prompt para hacer otras tareas. El comando que se corre en segundo plano (en “background”) puede volver a primer plano (“foreground”) con el comando *fg*.

El comando *fg* ejecuta un proceso en primer plano. Si no se le especifica el PID, traerá a primer plano el último proceso enviado a segundo plano.

Ejemplo

Intentaremos calcular el número π (pi) con 50.000 dígitos de precisión con la ayuda del comando *bc*. Para ello ejecutaremos

```
diego0 ~ # echo "scale=50000; 4*a(1)" | bc -l
```

Y comprobaremos que la máquina demora mucho en hacerlo y parece "colgarse" (aunque en realidad sólo está haciendo el cálculo y demorando mucho, si esperamos lo suficiente seguramete tendremos nuestro resultado). Si en otra consola ejecutamos el comando *top*, veremos que *bc* está consumiendo todo el CPU.

Para evitar esto, podemos ejecutar la tarea de recién pero en segundo plano, de la siguiente forma

```
diego0 ~ # echo "scale=50000; 4*a(1)" | bc -l &
```

Y después bash nos devolverá el prompt y podremos seguir ejecutando otros comandos. Una vez que *bc* termine su trabajo, escribirá el resultado del cálculo en la pantalla, lo que nos puede tomar por sorpresa, por lo que se recomienda correr tareas en segundo plano de la siguiente forma

```
diego0 ~ # echo "scale=50000; 4*a(1)" | bc -l > /root/pi.txt &
```

lo que enviará la salida del comando bc a un archivo y no a la pantalla y así evitamos sorpresas. Si en algún momento quisiéramos que *bc* continuara su ejecución en primer plano sólo tendríamos que ejecutar el comando *fg*.

4.14 . Niveles de arranque

Un nivel de arranque ("runlevel") se refiere a la forma en que opera el sistema de arranque en los sistemas UNIX System V (y por consiguiene Linux). Es un concepto un poco evasivo al principio, pero con la práctica se vuelve natural.

En Linux generalmente se usan siete niveles de arranque (de cero a seis), de los cuales por defecto el primero es el apagado del computador, el sexto es el rebuteo y del primero al quinto son los operativos, y difieren en los demonios que se levantan, los dispositivos que se montan, y otros. Generalmente los más bajos se usan para mantenimiento y recuperación de emergencia.

Nivel de ejecución	Nombre	Descripción
0	Alto	Cierre del sistema (apagado)
1	Monousuario	No configura la interfaz de red o los demonios de inicio, ni permite el logueo de otro usuario que no sea root. Se usa para pruebas y reparar problemas
2	Multiusuario	Multiusuario sin soporte de red

3	Multiusuario con soporte de red	Inicia el sistema normalmente
4		Libre (no usado)
5	Multiusuario gráfico	Similar al nivel de arranque 3 más *display manager*
6	Reinicio	Reinicio del sistema

Tabla 6: Niveles de arranque por defecto

Los niveles de arranque en cualquier distribución Linux son **completa y absolutamente personalizables**. La configuración de arranque está escrita en el *script de arranque* **/etc/inittab** y es leído una vez por el proceso *init* durante el buteo (aunque se puede forzar una relectura con el comando *telinit Q* o *telinit q* si es que se realizaron cambios en el mismo y queremos que surtan efecto en caliente). Puede pasarse a otro nivel de arranque con el comando *init* o *telinit* (son sinónimos) y el nievel de arranque al que queremos pasar. Para saber en qué nievel de arranque estamos podemos ejecutar el comando *runlevel*.

Para ver en detalle cómo se puede personalizar este archivo, ver *man inittab(5)*.

Las acciones dentro de /etc/inittab se definen de la siguiente forma:

```
<id>:<niveles_de_ejecución>:<acción>:<proceso>
```

donde:

- ***<id>*** es una secuencia única de uno a cuatro caracteres que identifica la entrada (hasta dos para mayor compatibilidad con sistemas viejos, no es necesario).
- ***<niveles_de_ejecución>*** es una lista de los niveles de arranque para los cuales la acción especificada se realizará. Puede contener varios caracteres para diferentes niveles de ejecución; por ejemplo, “123” significa que el proceso debería iniciarse para los niveles de ejecución 1, 2 y 3. Este campo se ignora cuando el campo <acción> toma cualquiera de los valores siguientes: *sysinit*, *boot* y *bootwait*.
- ***<acción>*** describe la acción que se tomará. Los valores más frecuentes que puede tomar este campo son:
  - **respawn**: el proceso se reiniciará cuando sea terminado (por ejemplo, *agetty*).
  - **wait**: el proceso se iniciará una vez cuando se entre en el nivel de ejecución e *init* esperará por su terminación.
  - **once**: el proceso se iniciará una vez cuando se entre en el nivel de ejecución.
  - **boot**: el proceso se iniciará durante el buteo del sistema.
  - **bootwait**: el proceso se iniciará una vez durante el buteo e *init* esperará su terminación (por ejemplo: */etc/rc*).
  - **off**: esto no hace nada.
  - **initdefault**: esta entrada especifica en qué nivel de ejecución se debería entrar después del buteo. Si no se especifica este campo *init* pedirá un nivel de ejecución en la consola. El campo <proceso> es ignorado.
  - **sysinit**: se ejecutará durante el buteo antes de cualquier entrada *boot* o *bootwait*.
  - **powerwait**: el proceso se ejecutará cuando se corte el suministro eléctrico.

Generalmente un proceso conectado al UPS con el que está enchufado el computador informa a i*nit* sobre esto. *init* esperará a que el proceso termine antes de continuar.

- **powerfail**: como el anterior, pero *init* no espera la terminación del proceso.
- **powerokwait**: el proceso se ejecutará tan pronto como *init* sea informado de que la alimentación ha sido restaurada.
- **powerfailnow**: el proceso se iniciará cuando *init* sea informado que la batería del UPS externo está casi vacía y que la alimentación está fallando (si es que el UPS externo y el proceso monitor sean capaces de detectar este estado).
- **ctrlaltdel**: el proceso se ejecutará cuando *init* reciba la señal SIGINT. Esto es, alguien en la consola presionó la combinación de teclas *ctrl+alt+del*.

➢ **<*proceso*>** es el proceso que se ejecutará.

Cuando se cambia de nivel de ejecución cualquier proceso que no esté especificado en el nuevo nivel de ejecución es matado por el proceso *init* con la señal SIGTERM primero y, si al cabo de un tiempo (habitualmente cinco segundos) no se terminó, es matado definitivamente con la señal SIGKILL.

4.15 . Particionamiento de discos en modo texto

La herramienta más común para este trabajo es *fdisk*, además de la más usada (más aún que los particionadores gráficos). Un comando más visual (con menús y otros) pero menos potente es *cfdisk*.

Descripción de la herramienta de particionamiento *fdisk*

Manipulador de tabla de particiones para Linux. La tabla de particiones se encuentra en el sector cero de los discos duros y describe las particiones que los componen.

El uso más común para este comando es:

```
diego0 ~ # fdisk <dispositivo_de_bloque>
```

donde <dispositivo_de_bloque> es el nombre de un dispositivo de bloque encontrado en el directorio **/dev/** y que representa un disco duro. Un ejemplo puede ser "/dev/sda".

Opciones:

➢ -b tamañodesector: especifica el tamaño de sector del disco, y "tamañodesector" puede valer 512, 1024 ó 2048. Se usa en kernels viejos o para sobreescribir las ideas del kernel.

➢ -l: lista la tabla de particiones de los dispositivos especificados y termina. Si no se especifican, usa los que encuentra en **/proc/partitions** (si existe).

Una vez en el prompt de fdisk, se tipea *m* y luego <Entrar> para pedir ayuda de todas las acciones posibles. Las descripciones de las acciones son tan claras y éstas tan precisas y atómicas que no vale la pena describirlas.

4.16 . Montaje de dispositivos

Para montar y desmontar dispositivos se usan los comandos mount y umount,

respectivamente. Los dispositivos se montan y desmontan usando *opciones de montaje*, que indican al sistema cómo tratar al sistema de archivos en los dispositivos.

Archivo /etc/fstab (File Sistems TABle)

En este archivo se detallan los dispositivos y sistemas de archivos y las opciones con las que se van a montar y desmontar, por defecto.

Las líneas que comienzan con el símbolo "numeral" ("#") son ignoradas (comentarios).

Cualquier otra línea se compone de seis campos separados por blancos (espacios o tabulaciones horizontales) de la siguiente forma:

- Sistema de archivo: nombre completo del dispositivo a montar.
- Punto de montaje: directorio del árbol jerárquico en el que se montará.
- Tipo: un tipo válido de sistema de archivo.
- Opciones: lista de opciones de montaje separada con comas.
- Dump: este campo es utilizado por la utilidad *dump* para decidir cuándo hacer un backup. Cuando está instalado (generalmente no se instala por defecto) dump crea un backup del sistema de archivos si está marcado con 1 y no lo hace si está marcado con 0.
- Pass: orden en el que fsck hará el chequeo. Los valores posibles son 0, 1 y 2. Cero significa que no será chequeado. Uno es la mayor prioridad y dos la menor.

Archivo /etc/mtab (Mounted file systems TABle)

Muestra los sistemas de archivos montados en un momento particular, en forma de campos separados por un blanco de la misma forma y orden que en /etc/fstab.

Comando mount

Monta un sistema de archivos. Opciones:

- -a: Montar todos los sistemas de archivos (del tipo dado) mencionados en /etc/fstab.
- -n: montar sin escribir en /etc/mtab.
- -t: especificar el tipo de sistema de archivo.
- -o: opciones de montaje separadas por comas.
- --bind: remontar un subárbol en otro lado (así es accesible desde ambos lugares).
- --move: mover un subárbol a otro lado.
- -v: modo verboso.

Comando umount

Desmonta un sistema de archivos. Opciones:

- -v: modo verboso.
- -n: desmontar sin escribir en /etc/mtab.
- -a: desmontar todos los sistemas de archivos descritos en /etc/mtab.
- -t: tipo de sistema de archivo.
- -O: opciones de montaje separadas por comas.
- *-f*: forzar el desmontado.

Opciones de montaje

Las opciones de montaje se sobreescriben a medida que son leídas. Un prefijo **no** indica que

lo opuesto a la opción.

- **async**: toda la E/S en el sistema de archivos debe hacerse en forma asincrónica.
- **atime**: actualizar el tiempo de acceso de cada inodo en cada acceso. Se usa *noatime* para un acceso más veloz al *spool* de noticias para acelerar servidores de noticias.
- **auto**: puede ser montado con la opción *-a* (en el caso del comando mount). Se montará automáticamente durante el buteo (en el caso del archivo /etc/fstab).
- **defaults**: usar las opciones por defecto: rw, suid, dev, exec, auto, nouser, y async.
- **dev**: interpretar los dispositivos especiales de bloque o de caracter dentro del sistema de archivos.
- **exec**: permitir la ejecución de binarios.
- **group**: permitir a un usuario ordinario montar el sistema de archivos si uno de sus grupos es el del dispositivo. Esta opción implica las opciones nosuid y nouid.
- **nodiratime**: no actualizar el tiempo de acceso en los inodos de directorios.
- **relatime**: actualizar el tiempo de acceso de los inodos relativamente al tiempo de modificación o cambio. El tiempo de acceso sólo se actualiza si el tiempo de acceso anterior fue hecho antes que el actual tiempo de modificación o cambio.
- **owner**: permitir a un usuario ordinario montar el sistema de archivos si es el propietario del dispositivo. Esta opción implica las opciones nosuid y nodev.
- **remount**: intentar remontar un sistema de archivos ya montado. No afecta el dispositivo ni el punto de montaje.
- **ro**: sólo lectura.
- **rw**: lecto-escritura.
- **suid**: permitir que funcionen los bits SUID o SGID.
- **sync**: toda la E/S en el sistema de archivos debe hacerse en forma sincrónica. En el caso de dispositivos con ciclos de escritura limitados (como algunos discos *flash*) esta opción puede ocasionar el acortamiento del ciclo de vida.
- **user**: permitir que un usuario ordinario monte el sistema de archivos. El nombre de este usuario se escribe en /etc/mtab para que pueda luego desmontarlo. Esta opción implica las opciones noexec, nosuid y nodev.
- **users**: permitir que cualquier usuario pueda montar y desmontar el sistema de archivos. Esta opción implica las opciones noexec, nosuid y nodev.

Tipos de sistemas de archivos soportados actualmente: adfs, affs, autofs, cifs, coda, coherent, cramfs, debugfs, devpts, efs, ext, ext2, ext3, hfs, hfsplus, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, nfs4, ntfs, proc, qnx4, ramfs, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, usbfs, vfat, xenix, xfs, xiafs.

5. Bash

5.1. Introducción

Más información sobre programación Bash **http://www.tldp.org/LDP/abs/html/** y **http://www.network-theory.co.uk/docs/bashref/index.html**.

Se puede personalizar Bash en los archivos **/etc/bash/bashrc**, **/etc/bash/bash_logout** y **/etc/bash/bash_login** que son, respectivamente, los *scripts* que ejecutará Bash al iniciarse, al terminar sesión y al iniciar sesión.

En el directorio personal de cada usuario es posible crear un script de Bash con el nombre corto de cualquiera de los anteriores archivos, precedidos por un punto (“.”), que se ejecutarán después de ejecutar los respectivos scripts que Bash encuentre en /etc/bash/.

Bash es sensible a la capitalización.

5.2. Variables y parámetros de entrada en scripts

En Bash, una variable es un lugar donde alojar datos. La recuperación (referenciación o expansión) de esos datos se llama *sustitución de variable*, y se logra anteponiendo el símbolo “pesos” (“$”) al nombre de la variable encerrado entre llaves (“{”, “}”). También puede aparecer sin las llaves (forma corta), pero en contextos dudosos las llaves guardan la salvedad.

Alcance

Las variables pueden tener alcance global (de ambiente) o local: las primeras (y sus respectivos valores) estarán disponibles para todos los shells; y, las segundas, sólo tendrán alcance en la ejecución de la instancia de Bash que las declara.

Tipo

Las variables pueden ser de cuatro tipos según su contenido:

- Variables de cadena
- Variables enteras
- Variables constantes (valga la contrariedad)
- Variables de arreglos

Creación de variables y asignación

Las variables se crean cuando son necesarias, cuando se las necesita. No se las declara solamente, sino que también se las define.

Con la palabra *export* antecediendo a la variable en declaración, decimos que ésta tendrá alcance global y no local.

Las asignaciones se hacen **sin espacios** entre la variable, el símbolo “asignación” (“=”) y el

valor asignado, de la siguiente forma:

```
diego0 ~ # export hola=”alcance global”
diego0 ~ # num=5
diego0 ~ # echo -e “${hola}.\nnum vale $num”
alcance global.
num vale 5
diego0 ~ # _
```

Variables de entorno especiales

- PS1: es el prompt, lo que escribe Bash cuando está listo para recibir órdenes.
- PS2: prompt secundario.
- USER: el usuario actualmente logueado.
- HOME: directorio personal del usuario logueado.
- SHELL: intérprete.
- TERM: tipo de terminal en la que se está ejecutando.
- PATH: dónde buscar ejecutables. A diferencia del intérrete de DOS, en Linux no se buscan ejecutables dentro del directorio actual, sólo en el PATH, lo que brinda un poco más de seguridad. Para ejecutar archivos en el directorio actual sólo se antepone a su nombre corto “./” (bastante lógico, ¿no?).

Parámetros de entrada

Los parámetros en un *script* de Bash se reciben en forma posicional y se utilizan los enteros positivos para numerarlos y expandir sus resultados. Por ejemplo, “$1” expande al primer parámetro recibido.

Para moverse entre los parámetros se usa el comando interno *shift* seguido o no de un número positivo (uno por defecto). Este comando hace un desplazamiento de la lista de parámetros, descartando el primero, de tal forma que $2 es ahora $1, etc. Si el número especificado es más que uno, se desplaza esa cantidad de veces la lista de parámetros, de modo que si se escribe *shift 3*, entonces se descartan los tres primeros parámetros y $4 es ahora accesible a través de $1 y así sucesivamente.

Existen parámetros especiales, estos son:

- #: cantidad de parámetros que hay actualmente.
- 0 (cero): nombre del shell o del script en ejecución.

Existen varios otros, lo invitamos a investigar.

5.3 . Sintaxis

Scripts

Un script de Bash es un archivo de texto ejecutable que contiene rutinas que puede ejecutar el intérprete de Bash.

Las líneas que comienzan con un símbolo “numeral” se ignoran y sirven como comentarios.

La primera línea comienza es similar a la siguiente:

```
#!/bin/sh
```

lo que le indica al shell que usemos (sea o no Bash) que el intérprete que entenderá propiamente (ejecutará) este script es **/bin/bash**. Si escribiéramos un script de Python, por ejemplo, entonces escribiríamos el nombre del intérprete de Python en esta línea.

Signos de puntuación

- Punto y coma o "semicolon" (";"): se utiliza para separar tareas. Ejemplo:

```
diego0 ~ # comando0 arg0; comando1 arg1; comando2 arg2
```

- Comillas simples, apóstrofes o "simple quotes" ("'"): guarda el valor literal de cada caracter contenido entre ellas. No puede haber una comilla simple entre dos comillas simples, ni siquiera precedida por una barra invertida.
- Comillas dobles (""): guarda el valor literal de cada caracter contenido entre ellas excepto para los caracteres "$", "`", "\" y, cuando está habilitada *history expansion*, "!". El caracter "\" mantiene su significado especial sólo si está seguido de cualquiera de los siguientes caracteres: "$", "", "`", "\" o "n" (nueva línea). Estas combinaciones **(*secuencias de escape*)** se convierten en los respectivos caracteres que subsiguen a la barra invertida. Si el caracter al que precede la barra invertida no es ninguno de los mencionados, la barra invertida no se escapa ningún caracter.
- Acentos graves, "back-ticks" o "back-quotes" ("`"): se utiliza para recoger la salida estándar de un comando ("command substitution"). El resultado se lo puede recoger en una variable. También se puede hacer esto encerrando el comando entre paréntesis y anteponiéndole el símbolo "pesos". Ejemplo:

```
diego0 ~ # hola=$(echo Estamos en `uname -s`)
diego0 ~ # echo $hola
Estamos en Linux
```

- Llaves o "braces" ("{", "}"): cobinación de elementos. Dentro de las llaves se especifica una lista separada por comas de cadenas de caracteres (entre comillas si se quieren usar comas y/o llaves dentro de esas cadenas) que se usarán en forma *combinatoria* con el resto de una cadena de caracteres. Si dentro de una cadena de caracteres aparece más de una de estas estructuras, las *combinaciones* se resuelven de izquierda a derecha. Esto se conoce como expansión de llaves. Ejemplos:

```
diego0 ~ # echo {one,two,red,blue}fish rock it!
onefish twofish redfish bluefish rock it!
diego0 ~ # echo "{"{0,1}{a,b,c}{","","{"}"} -" ¿confuso?
{0a,} - {0a{} - {0b,} - {0b{} - {0c,} - {0c{} - {1a,} - {1a{} - {1b,}
- {1b{} - {1c,} - {1c{} - ¿confuso?
```

5.3.1 . Estructuras de control

Comando if

```
if <tests>; then
  <comandos>;
[elif <mas_tests>; then
  <mas_comandos>;]
[else <comandos_alternativos>;]
fi
```

Se ejecuta <tests> y, si su estado de retorno es cero, se ejecuta <comandos>, sino cada <mas_tests> de la lista de los *elif* se ejecuta en su debido turno en busca del que devuelva su estado de retorno cero para ejecutar su correspondiente <mas_comandos> y luego salir del comando *if*. Si ningún <mas_tests> satisface esta condición, se ejecuta <comandos_alternativos>. El estado de retorno es el estado de salida del último comando ejecutado, o cero si ninguna condición fue verdadera y no se especificó *else*.

Comando case

```
case <palabra> in
  [ [(] <patrón> [| <patrón>]...) <comandos> ;;]
  ...
esac
```

Este comando ejecutará selectivamente la lista de comandos <comandos> correspondiente al primer patrón <patron> que satisfaga <palabra>. El caracter "|" se usa para separar diferentes patrones, y, si se especifica más de uno, se los encierra entre paréntesis.

Una lista de patrones y una lista de comandos (en este sentido) se conoce como *cláusula*, y cada cláusula termina con ";;". <palabra> y cada <patrón> son interpretados antes de intentar la búsqueda de patrón.

El estado de retorno es cero si no hubo coincidencias, y si la hubo es el estado de salida del último comando ejecutado.

5.3.2 . Bucles

Bucle until

```
until <tests>; do
  <comandos>;
done
```

Ejecuta <comandos> mientras <tests> tengan un valor de estado de retorno diferente de cero. El estado de retorno el estado de salida del último comando ejecutado o cero si <comandos> no se ejecutó nunca.

Bucle while

```
while <tests>; do
  <comandos>;
done
```

Igual que *until*, sólo que se ejecuta <comandos> mientras <tests> tenga un estado de retorno cero.

Bucle for

```
for <nombre> [in <palabras> ...]; do
  <comandos>;
done
```

Se expande <palabras>. Si no se especifica <palabras>, es como si se especificara "$@", es decir, la lista de parámetros.

Se ejecuta sucesivamente <comandos> tantas veces como palabras haya en <palabras>, cada vez variando <nombre> de tal forma que va tomando como valor cada palabra en <palabras>. Si <palabras> es un directorio, entonces <nombre> tomará sucesivamente los nombres cortos de los archivos en el directorio.

El estado de retorno es el estado de salida del último comando ejecutado o cero si nunca se ejecuta <comandos> (es decir, si no hay palabras en <palabras>).

La siguiente es una sintaxis alternativa:

```
for (( <expr1> ; <expr2> ; <expr3> )) ; do
  <comandos>;
done
```

La explicación de esta forma debería resultar obvia para quien tiene una noción mínima de programación.

5.3.3 . Comando test

Chequea tipos de archivo y compara valores. Se escribe *test* o *[* y se termina con *]* en ese caso.

Se utiliza en *if*, *case*, etc. para hacer tests.

```
[ EXPR ]
```

EXPR es veradadera.

```
[ ! EXPR ]
```

EXPR es falsa.

```
[ EXPR1 -a EXPR2 ]
```

EXPR1 y EXPR2 son veradaderas.

```
[ EXPR1 -o EXPR2 ]
```

Cualquiera, EXPR1 o EXPR2, es veradadera.

```
[ -n CADENA ]
```

La longitud de CADENA no es cero.

```
[ -z CADENA ]
```

La longitud de CADENA es cero.

```
[ CAD1 = CAD2 ]
```

Las cadenas CAD1 y CAD2 son iguales.

```
[ CAD1 != CAD2 ]
```

Las cadenas CAD1 y CAD2 son diferentes.

```
[ ENT1 -eq ENT2 ]
```

Estos enteros son iguales.

```
[ ENT1 -ge ENT2 ]
```

ENT1 >= ENT2.

```
[ ENT1 -gt ENT2 ]
```

ENT1 > ENT2.

```
[ ENT1 -le ENT2 ]
```

ENT1 <= ENT2.

```
[ ENT1 -lt ENT2 ]
```

ENT1 < ENT2.

```
[ ENT1 -ne ENT2 ]
```

Estos enteros son diferentes.

```
[ -d ARCH ]
```

El archivo ARCH existe y es un directorio.

```
[ -e ARCH ]
```

El archivo ARCH existe.

```
[ -f ARCH ]
```

El archivo ARCH existe y es un archivo regular.

```
[ -h ARCH ]
[ -L ARCH ]
```

El archivo ARCH existe y es un symlink.

```
[ -s ARCH ]
```

El archivo ARCH existe y su tamaño es mayor que cero.

```
[ -r ARCH ]
```

El archivo ARCH existe y se permite leerlo.

```
[ -w ARCH ]
```

El archivo ARCH existe y se permite escribirlo.

```
[ -x ARCH ]
```

El archivo ARCH existe y se permite ejecutarlo.

5.3.4 . Salida

Para fines prácticos, un comando que con un estado de salida cero significa que tuvo éxito, y uno diferente de cero significa que no lo tuvo.

En cualquier parte de un script puede escribirse la sentencia *exit*, que termina el script.

6 . Instalación de software externo a la distribución

6.1 . Tarballs

Muchas veces en Linux vamos a encontrar programas que no son instalables desde la herramienta de instalación de nuestra distribución, tal puede ser el caso de *Apache*, *VirtualBox*, etc., y tenemos que instalar ese paquete "a mano". Esto significa descargar las fuentes (el código fuente) desde Internet y compilarlo en nuestro equipo. Habitualmente, encontraremos que las fuentes están en un sólo archivo (el que descargamos de internet) llamado tarball.

Éste es un formato ampliamente difundido (especialmente en los sistemas UNIX) que se identifica con la extensión *.tar*. Estos archivos son capaces de almacenar archivos y directorios dentro de sí mismos, de modo que quedan "empaquetados". Es tan extendido su uso que podemos arriesgarnos a decir que cualquier plataforma cuenta con al menos una versión de un manipulador de archivos tar, ya sea con un GUI o en línea de comandos, freeware, open source o software propietario.

También es común que no se trate de un tarball simplemente, sino de uno comprimido por diferentes métodos, como GZIP o BZIP2. Los tarballs de por sí no están comprimidos, sino que compilan en sí varios archivos; se usan varios métodos, como los antes mencionados, para comprimir su contenido, y de esa forma se tienen archivos *.tar.gz* y *.tar.bz2*.

Para la completa manipulación de archivos en los formatos TAR, ZIP, 7Z, GZIP y BZIP2 bajo plataformas Windows recomendamos el programa "7-zip", que es de código abierto (con licencia GNU LGPL) y de muy buen rendimiento y puede descargárselo gratuitamente desde su sitio oficial **http://www.7-zip.org/**. Esto quiere decir que puede disfrutarse de sus bondades sin pagar por ellas y sin ***molestos avisos*** pidiéndonos que registremos nuestro producto.

En Linux el comando más importante para la manipulación de tarballs es *tar*, y a continuación brindamos una breve descripción.

tar es la versión GNU de la utilidad de archivación tar. Está diseñado para guardar y para recoger datos de tarballs.

Letras funcionales (debe usarse una de las siguientes opciones):

- *[-]A, --catenate, --concatenate*: agregar archivos tar a un tarball.
- *[-]c, --create*: crear un nuevo tarball.
- *[-]d, --diff, --compare*: comparar entre un tarball y un sistema de archivos.
- *[-]r, --append*: agregar archivos al final del tarball.
- *[-]t, --list*: listar el contenido de un tarball.
- *[-]u, --update*: sólo agregar archivos a un tarball si son más nuevos que los existentes.
- *[-]x, --extract, --get*: extraer archivos de un tarball.
- *--delete*: eliminar del tarball.

Opciones comunes:

- *-C, --directory DIR*: cambiar al directorio DIR.
- *-j, --bzip2*: filtrar el tarball con bzip2.
- *-f --file [HOSTNAME:]F*: usar el archivo o dispositivo F (por defecto "-", que quiere decir stdin/stdout). Sirve para decir que trabajamos con un archivo.
- *-p, --preserve-permissions*: extraer toda la información de protección.

- *-v, --verbose*: modo verboso.
- *-z, --gzip, --ungzip*: filtrar el tarball con gzip.

6.2 . make

Una vez descomprimido el tarball y situados en el directorio de fuentes, tendremos que seguir tres pasos que más o menos serán los siguientes:

```
diego0 ~ # ./configure && make && make install
```

El último de los cuales es a veces opcional. Dicho esto a secas, queda mucho por explicar, como por ejemplo que el último paso debe ejecutarse como root.

El comando

El principal y más popular uso del comando make es la instalación de paquetes de software y su actualización y es este propósito el que se va a aprovechar en este manual.

El comando make utiliza un archivo, el *Makefile*, que le indica qué piezas del programa necesitan ser compiladas y cuáles recompiladas. Una característica ventajosa de make es que podemos empezar a instalar un programa, cortar la instalación y luego reanudarla en cualquier momento desde el punto en que había quedado.

El Makefile es un archivo de texto que le da a make un panorama del sistema que tenemos y de su contexto, para poder instalar propiamente los programas. Generalmente este archivo es generado por otro archivo, un script, llamado *configure*, que lo podremos encontrar en el directorio de fuentes. Una opción que podemos darle a este script es dónde queremos que se instale nuestro programa (seguramente querremos que sea en **/usr/local/**, aunque también puede ser en **/opt/** u otro lugar de nuestra preferencia) escribiendo "**--prefix=/usr/local/miapp/**". Esto quiere decir que todos los archivos, librerías, includes, etc. de *miapp* se instalarán en ese directorio y no se mezclará con los demás archivos de mimo tipo, facilitando la posterior supresión de la aplicación.

El script posee también una opción *--help* que nos da ayuda sobre opciones de instalación específicas para el paquete que estamos instalando

Una vez ejecutado este script y, si todo anda bien, podemos ejecutar finalmente *make*, que se encargará de llamar a los compiladores (cc, gcc, g++, etc.) necesarios y dejará al programa funcional.

El último paso crea el archivo man, info (si es que están disponibles) y hace que el binario sea accesible desde el path, es decir, que en adelante podrá ejecutarse escribiendo en la consola sólo el nombre corto y no el largo (con toda la ruta al binario). En algunos casos este paso es importante porque instala dependencias y librerías.

Generalmente en el mismo directorio de fuentes podremos encontrar un archivo de nombre *INSTALL* que nos dará información y nos ayudará con la instalación.

make clean y *make uninstall*

En forma opcional después de la instalación de un programa puede ejecutarse el comando "make clean", que liberará espacio en el disco al eliminar archivos colaterales a la compilación e

instalación del mismo. Esto es especialmente útil después de una instalación fallida (o incluso después de una exitosa), por ejemplo, al ejecutar “make install” se descubre que no se cubrió una dependencia (raros casos) o que debemos cambiar manualmente cierta configuración del sistema. Luego de ejecutar “make clean” se puede reanudar la instalación desde el principio.

El comando “make uninstall” es una utilidad provista por *algunos* autores de software para asegurar una desinstalación exitosa y sin riesgos (nótese el énfasis en “algunos”).

6.3 . Ejemplo: instalación de Apache

Paso I: descargar las fuentes

Puede hacérselo desde **http://archive.apache.org/dist/httpd/** en el navegador en línea de comandos links (habrá que instalarlo si no viene con nuestra distribución). Para ello primero nos situamos en el directorio donde queremos descargarlas.

```
diego0 ~ # cd /opt/apache/
diego0 apache # links http://archive.apache.org/dist/httpd/
```

Bajamos con la tecla “flecha abajo” hasta dar con el link al archivo más reciente (en este caso “apache_1.3.9.tar.gz”) y presionamos <Entrar> y aceptamos el diálogo de descarga.

Paso II: descomprimir

```
diego0 apache # tar xfz apache_1.3.9.tar.gz
diego0 apache # cd apache_1.3.9
diego0 apache_1.3.9 # ls -lah --color=always
total 236
drwxr-xr-x  8  161 cdrom  4096 Aug 16  1999 ./
drwxr-xr-x  3 root root   4096 Mar 30 10:17 ../
-rw-r--r--  1  161 cdrom 12957 Mar 31  1999 ABOUT_APACHE
-rw-r--r--  1  161 cdrom  3679 Mar 22  1999 Announcement
-rw-r--r--  1  161 cdrom 27293 Aug  9  1999 INSTALL
-rw-r--r--  1  161 cdrom 33398 Aug  7  1999 KEYS
-rw-r--r--  1  161 cdrom  2848 Jan  1  1999 LICENSE
-rw-r--r--  1  161 cdrom 26414 Aug 13  1999 Makefile.tmpl
-rw-r--r--  1  161 cdrom  2046 Apr  1  1998 README
-rw-r--r--  1  161 cdrom  3132 Mar 19  1999 README.NT
-rw-r--r--  1  161 cdrom 11687 Feb  7  1999 README.configure
-rw-r--r--  1  161 cdrom   331 Sep 21  1998 WARNING-NT.TXT
drwxr-xr-x  2  161 cdrom  4096 Aug 16  1999 cgi-bin/
drwxr-xr-x  2  161 cdrom  4096 Aug 16  1999 conf/
-rw-r--r--  1  161 cdrom  4701 Jul 29  1999 config.layout
-rwxr-xr-x  1  161 cdrom 52983 Aug 14  1999 configure*
drwxr-xr-x  3  161 cdrom  4096 Aug 16  1999 htdocs/
drwxr-xr-x  3  161 cdrom  4096 Aug 16  1999 icons/
drwxr-xr-x  2  161 cdrom  4096 Aug 16  1999 logs/
```

```
drwxr-xr-x 11  161 cdrom  4096 Aug 16  1999 src/
diego0 apache_1.3.9 # _
```

Paso III: leer el archivo INSTALL

Se recomienda siempre leer los archivos que vienen con los paquetes de software.

Paso IV: compilar las fuentes

```
diego0 apache_1.3.9 # ./configure --prefix=/usr/local/apache_1.3.9/
Configuring for Apache, Version 1.3.9
 + using installation path layout: Apache (config.layout)
Creating Makefile
Creating Configuration.apaci in src
Creating Makefile in src
 + configured for Linux platform
 + setting C compiler to gcc
 + setting C pre-processor to gcc -E
 + checking for system header files
 + adding selected modules
 + checking sizeof various data types
 + doing sanity check on compiler and options
Creating Makefile in src/support
Creating Makefile in src/regex
Creating Makefile in src/os/unix
Creating Makefile in src/ap
Creating Makefile in src/main
Creating Makefile in src/lib/expat-lite
Creating Makefile in src/modules/standard
diego0 apache_1.3.9 # make
     .
     .
     .
diego0 apache_1.3.9 # make install
     .
     .
     .
diego0 apache_1.3.9 # _
```

Opcional: probar la funcionalidad del programa

Para hacerlo lo corremos al servidor ejecutando

```
diego0 apache_1.3.9 # /usr/local/apache_1.3.9/bin/apachectl start
/usr/local/apache_1.3.9/bin/apachectl start: httpd started
diego0 apache_1.3.9 # _
```

y luego lo probamos con el navegador links

```
diego0 apache_1.3.9 # 127.0.0.1:80
```

Lo que nos debería mostrar una página de bienvenida.

Atención

Puede ser que no pueda levantarse Apache si no puede encontrar el nombre del host local, en cuyo caso

```
diego0 apache_1.3.9 # /usr/local/apache_1.3.9/bin/apachectl start
httpd: cannot determine local host name
Use the ServerName directive to set it manually.
/usr/local/apache_1.3.9/ bin/apachectl start: httpd could not be
started
```

y tendríamos que editar el archivo de configuración de Apache, que en este caso es /usr/local/apache_1.3.9/conf/httpd.conf y descomentar la línea en la que se encuentra la directiva ServerName, que en este caso es

```
#ServerName diego0.fortix.com.ar
```

Para descomentar esa línea sólo quitamos el símbolo "numeral" ("#").

Opcional: crear un init-script para la aplicación

Si se quiere que la aplicación se inicie con el sistema habrá que crear un script que se ejecute al entrar en el nivel de arranque por defecto (o en los que se desee).

El sistema de scripts de arranque de System V perdura todavía en muchas distribuciones, como Debian y derivados, y por ello se van a explicar a continuación, que pretende ser una traducción al español resumida (con modificaciones a discreción) del contenido de **http://www.debian.org/doc/debian-policy/ch-opersys.html#s-sysvinit**. Otras distribuciones, como Gentoo y derivados, utilizan el sistema descrito en **http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=2&chap=4**.

6.4 . Sistema de init scripts de distribución Debian y derivadas

El directorio **/etc/init.d/** contiene los scripts ejecutados por el proceso *init* durante el buteo y cuando el nivel de arranque se cambia.

Hay al menos dos maneras diferentes, si bien equivalentes, de manejar estos scripts, de las cuales sólo se explicará una (el método de los symlinks) por razones de simpleza. Sin embargo, el administrador de estos scripts no debería asumir que este método está siendo usado, y cualquier manipulación automatizada de éstos debería realizarla con el comando *update-rc.d*. Para obtener información sobre el otro método se puede consultar la documentación del paquete *file-rc* (que lo implementa).

Estos scripts son referenciados por symlinks en el directorio **/etc/rcS.d** (para los scripts de buteo) y en los directorios **/etc/rcN.d** (N es el nivel de arranque), donde init busca los scripts

que debe ejecutar.

Los nombres de los links tienen la forma **[S|K]nnScript**, donde nn es un número de dos dígitos y Script es el nombre del script (que debería ser el mismo que en **/etc/init.d/**). Este número sirve para indicar el orden de ejecución de los scripts: los scripts con un número más bajo se ejecutan primero que los de número más alto.

Cuando *init* cambia el nivel de ejecución primero ejecuta los scripts referenciados en el directorio **/etc/rcN.d/** (siendo N el runlevel al que se está cambiando) que comienzan con K (kill o "matar"), cada uno con un único parámetro, *stop*, y luego ejecuta los que comienzan con S (start o "iniciar") con *start* como parámetro único.

En los runlevels 0 (apagado) y 6 (reinicio) se ejecutan primero los scripts referenciados con una K y luego los referenciados con una S, pero a todos se les pasa el mismo argumento único *stop*.

6.4.1 . Cómo escribir los scripts

Los paquetes que incluyen demonios deberían incluir los scripts bajo el nombre **/etc/init.d/*paquete***, y deberían aceptar uno de los siguientes argumentos:

- **start**: inicia el servicio.
- **stop**: detiene el servicio.
- **restart**: detiene y reinicia el servicio si ya está corriendo, sino lo inicia.
- **reload**: provoca que la configuración del servicio se recargue sin realmente detenerlo reiniciarlo.
- **force-reload**: provoca que la configuración del servicio se recargue si el servicio lo soporta, sino lo reinicia.

Las opciones *start*, *stop*, *restart* y *force-reload* deberían ser soportadas por todos los scripts en **/etc/init.d/**. La opción *reload* es opcional.

Los scripts de **/etc/init.d/** deben asegurar que se comportarán en forma sensible (por ejemplo, devolviendo *éxito* y no iniciando múltiples copias de un servicio) si se invoca con *start* cuando el servicio ya está corriendo, o con *stop* cuando no lo está. La mejor forma de hacer esto es usar el comando *start-stop-daemon* con la opción *--oknodo*. Para más información sobre este comando consultar el comando *man*.

6.4.2 . Creación de los links

Para crear los links se utiliza el comando *update-rc.d*, que tiene la siguiente sintaxis

```
update-rc.d [-n] nombre start|stop NN runlevel runlevel ...  .  start|
stop NN runlevel runlevel ...  . ...
```

6.4.3 . Ejemplo

Se provee junto con estas distribuciones el archivo **/etc/init.d/skeleton**, un "esqueleto" de init script para que sea más fácil crear los propios.

A partir de éste se construyó el siguiente init script, para ejecutar el demonio http de apache cuando se inicie el sistema.

```
diego0 ~ # vi /etc/init.d/apache
     .
     .
     .
#! /bin/sh
### BEGIN INIT INFO
# Provides:          apache
# Required-Start:    $local_fs $remote_fs
# Required-Stop:     $local_fs $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      S 0 1 6
# Short-Description: Apache httpd
# Description:       Apache httpd
### END INIT INFO

PATH=/usr/sbin:/usr/bin:/sbin:/bin
DESC="Apache httpd"
NAME=apache
DAEMON=/usr/local/apache_1.3.9/bin/apachectl
SCRIPTNAME=/etc/init.d/${NAME}

# Function that starts the daemon/service
do_start()
{
        $DAEMON start
        return 0
}

# Function that stops the daemon/service
do_stop()
{
        $DAEMON stop
        return 0
}

# Function that sends a SIGHUP to the daemon/service
do_reload() {
        $DAEMON restart
        return 0
}

case "$1" in
  start)
        do_start
        ;;
  stop)
        do_stop
        ;;
  restart|force-reload)
```

```
        do_reload
        ;;
  *)
        echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}"
>&2
        exit 3
        ;;
esac
     .
     .    SALIR GUARDANDO LOS CAMBIOS
     .
diego0 ~ # update-rc.d apache start 51 S .
diego0 ~ # _
```

6.5 . Configuración y compilación del kernel de Linux

Linux es el kernel o núcleo del SO libre GNU/Linux (llamado generalmente Linux), un clon del SO UNIX escrito desde cero por Linus Torvalds, y creció y se desarrolló con la ayuda de hackers de todo el mundo a través de Internet.

El kernel de Linux (de GNU/Linux, Linux en adelante) puede descargarse de su sitio oficial (ya mencionado con anterioridad en “3.1 . Cómo conseguir Linux”), configurarse y compilarse cuando y como se quiera, sin importar el equipo que se tenga. Es decir, en una máquina con plataforma de 32 bits puede compilarse un kernel para 64 bits sin ningún problema (claro que en esa máquina no va a funcionar correctamente).

El kernel de Linux es un núcleo monolítico con un sistema de módulos ejecutables en tiempo de ejecución, que le brinda algunas de las ventajas de los micronúcleos. Es un error confundir este tipo de núcleo con los núcleos híbridos, que usan conceptos de arquitectura tanto del diseño monolítico como del micronúcleo.

Descompresión

Ejecutamos en una terminal el siguiente comando, suponiendo que el tarball con las fuentes del kernel haya sido descargado en el directorio /usr/src/, y que la versión descargada sea la indicada. Se recomienda este directorio pues algunos paquetes de software pueden necesitar de las fuentes del kernel para alguna tarea particular, y por defecto buscarán en este directorio.

```
diego0 ~ # cd /usr/src
diego0 src # tar xjf linux-2.6.29.tar.bz2
diego0 src # l
drwxr-xr-x  3 root root 4,0K 2009-04-07 10:02 .
drwxr-xr-x 21 root root 4,0K 2009-03-31 04:45 ..
drwxr-xr-x 22 root root 4,0K 2009-03-23 20:12 linux-2.6.29
-rw-r--r--  1 root root  54M 2009-04-07 10:02 linux-2.6.29.tar.bz2
diego0 src # ln -s linux-2.6.29 linux
diego0 src # cd linux-2.6.29
```

```
diego0 linux-2.6.29 # l
total 412K
drwxr-xr-x 22 root root 4,0K 2009-03-23 20:12 .
drwxr-xr-x  3 root root 4,0K 2009-04-07 10:02 ..
drwxr-xr-x 23 root root 4,0K 2009-03-23 20:12 arch
drwxr-xr-x  2 root root 4,0K 2009-03-23 20:12 block
-rw-r--r--  1 root root  19K 2009-03-23 20:12 COPYING
-rw-r--r--  1 root root  92K 2009-03-23 20:12 CREDITS
drwxr-xr-x  3 root root 4,0K 2009-03-23 20:12 crypto
drwxr-xr-x 82 root root 4,0K 2009-03-23 20:12 Documentation
drwxr-xr-x 84 root root 4,0K 2009-03-23 20:12 drivers
drwxr-xr-x 23 root root 4,0K 2009-03-23 20:12 firmware
drwxr-xr-x 65 root root 4,0K 2009-03-23 20:12 fs
-rw-r--r--  1 root root  867 2009-03-23 20:12 .gitignore
drwxr-xr-x 25 root root 4,0K 2009-03-23 20:12 include
drwxr-xr-x  2 root root 4,0K 2009-03-23 20:12 init
drwxr-xr-x  2 root root 4,0K 2009-03-23 20:12 ipc
-rw-r--r--  1 root root 2,4K 2009-03-23 20:12 Kbuild
drwxr-xr-x  6 root root 4,0K 2009-03-23 20:12 kernel
drwxr-xr-x  6 root root 4,0K 2009-03-23 20:12 lib
-rw-r--r--  1 root root 4,0K 2009-03-23 20:12 .mailmap
-rw-r--r--  1 root root 108K 2009-03-23 20:12 MAINTAINERS
-rw-r--r--  1 root root  54K 2009-03-23 20:12 Makefile
drwxr-xr-x  2 root root 4,0K 2009-03-23 20:12 mm
drwxr-xr-x 46 root root 4,0K 2009-03-23 20:12 net
-rw-r--r--  1 root root  17K 2009-03-23 20:12 README
-rw-r--r--  1 root root 3,1K 2009-03-23 20:12 REPORTING-BUGS
drwxr-xr-x  7 root root 4,0K 2009-03-23 20:12 samples
drwxr-xr-x 12 root root 4,0K 2009-03-23 20:12 scripts
drwxr-xr-x  5 root root 4,0K 2009-03-23 20:12 security
drwxr-xr-x 20 root root 4,0K 2009-03-23 20:12 sound
drwxr-xr-x  2 root root 4,0K 2009-03-23 20:12 usr
drwxr-xr-x  3 root root 4,0K 2009-03-23 20:12 virt
diego0 linux-2.6.29 # _
```

6.5.1 . Configuración manual

Configurar el kernel de Linux puede parecer tedioso y difícil, pero después de hacerlo unas cuantas veces puede llegar a ser automático y hasta entretenido. Eso sí, para hacerlo hay que tener plena consciencia del hardware del equipo, y una herramienta que facilitará esto será el comando *lspci*, descrito con anterioridad.

Por ello, es conveniente usar al menos dos terminales para configurar el kernel: la primera para la configuración propiamente dicha, y otra para ejecutar comandos que nos brinden información sobre nuestro sistema.

Existen varias formas de configurar el kernel provistas dentro del mismo tarball, detalladas en el archivo README; la que se usará es *menuconfig*.

```
diego0 linux-2.6.29 # make menuconfig
```

Esto creará una interfaz gráfica en la terminal. Con las teclas “flecha izquierda” y “flecha derecha” se desplaza entre las posibles acciones: Select (seleccionar), Exit (salir) y Help (ayuda). Las teclas “flecha arriba” y “flecha abajo” desplaza el foco entre los ítems.

Escribiendo un símbolo “interrogación cerrado” (“?”) mostrará ayuda pertinente en todo momento, ya sea sobre el ítem enfocado o sobre el manejo de la interfaz.

Algunas características del kernel pueden estar desactivadas (no formarán parte del kernel), otras activadas como *módulos* y otras activadas para ser construidas *dentro del kernel*.

Cada vez que un kernel se carga durante el buteo, todas las características que fueron construidas *dentro del kernel* son cargadas: estas características forman parte indivisible del kernel, compiladas como un todo dentro de la imagen final del kernel. Por otro lado, las características construidas como módulos serán compiladas en archivos individuales y sus respectivos nombres tendrán la extensión “.o” o “.ko”, y estarán alamacenados en los directorios contenidos en **/lib/modules/<kernel>/**, donde <kernel> es la versión del kernel, y puede ser algo así como “2.6.24-gentoo-r8” (resultando la ruta **/lib/modules/2.6.24-gentoo-r8/**) o “2.6.24-23-generic” (**/lib/modules/2.6.24-23-generic/**). Estas características, una vez compiladas, se llaman *módulos* (o *módulos del kernel*) y deben ser cargados para ser ejecutadas; puede hacérselo manualmente con el comando *modprobe* y pasándole como argumento el nombre del módulo (que no es más que el nombre del archivo sin la extensión “.ko” ó “.o”) o automáticamente escribiendo el nombre del módulo en un archivo, cuyo nombre variará según la distribución: **/etc/modules** para *Ubuntu 8.04.2* y **/etc/modules.autoload.d/kernel-<versión_corta>** (<versión_corta> puede ser “2.4” o “2.6”, por ejemplo) para *Gentoo 2008.0*.

Los módulos del kernel son cargados una vez ya arrancado el sistema, una vez ya cargado el kernel mismo. Los módulos son extensiones del kernel, se utilizan para añadirle funcionalidades que queremos que tenga pero que no queremos que estén dentro del mismo para que sea más ligero.

Los módulos pueden ser cargados y descargados cuando y como queramos. Para descargar un módulo podemos usar el comando *rmmod* o el comando *modprobe -r*, pasándoles como argumento el nombre del módulo a descargar.

¿Qué necesita tener el kernel?

Fundamentalmente deben estar dentro del kernel los controladores vitales, como por ejemplo el controlador para el disco en el que se encuentra montado el directorio raíz (puede ser para PATA, SATA, etc.).

- **Familia de procesadores**

Para averiguar cuál es la familia de procesadores al que pertenece el nuestro podemos ejecutar

```
diego0 ~ # cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      :               Intel(R) Pentium(R) 4 CPU 3.00GHz
stepping        : 3
cpu MHz         : 2999.924
cache size      : 2048 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 5
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe
syscall nx lm constant_
tsc pebs bts pni monitor ds_cpl est cid cx16 xtpr
bogomips        : 6005.48
clflush size    : 64
cache_alignment : 128
address sizes   : 36 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 15
model           : 4
model name      :               Intel(R) Pentium(R) 4 CPU 3.00GHz
stepping        : 3
cpu MHz         : 2999.924
cache size      : 2048 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 5
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe
syscall nx lm constant_
tsc pebs bts pni monitor ds_cpl est cid cx16 xtpr
bogomips        : 5999.84
clflush size    : 64
```

```
cache_alignment : 128
address sizes   : 36 bits physical, 48 bits virtual
power management:
diego0 ~ # _
```

En sistemas multi-núcleo cada núcleo cuenta como un procesador.

y comprobamos que es un Pentium 4, por lo que en la configuración del kernel seleccionamos

```
▶▶ Processor type and features
     Processor family (Intel P4 / older Netburst based Xeon)
```

Si tiene un CPU Intel que soporte HyperThreading™ (como el del ejemplo), o si se tiene un sistema multi-CPU, se debe activar *Symmetric multi-processing support*

```
▶▶ Processor type and features
     [*] Symmetric multi-processing support
```

Si se dispone de más de 4GB de RAM y un CPU de arquitectura de 32bits es necesario activar *High Memory Support (64G)*.

```
▶▶ Processor type and features
     ▶▶ High Memory Support (4GB)
          (X) 64GB
```

- **Dispositivos de disco**

Ejecute el programa *lspci* para saber sobre las interfaces de disco de que dispone el sistema

```
diego0 ~ # lspci
00:00.0 Host bridge: Intel Corporation 82915G/P/GV/GL/PL/910GL Memory
Controller Hub (rev 04)
00:01.0 PCI bridge: Intel Corporation 82915G/P/GV/GL/PL/910GL PCI
Express Root Port (rev 04)
00:02.0 VGA compatible controller: Intel Corporation 82915G/GV/910GL
Integrated Graphics Controller (rev 04)
00:1b.0 Audio device: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) High Definition Audio Controller (rev 03)
00:1c.0 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) PCI Express Port 1 (rev 03)
00:1c.1 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) PCI Express Port 2 (rev 03)
00:1c.2 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) PCI Express Port 3 (rev 03)
00:1c.3 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
```

```
Family) PCI Express Port 4 (rev 03)
00:1d.0 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) USB UHCI #1 (rev 03)
00:1d.1 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) USB UHCI #2 (rev 03)
00:1d.2 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) USB UHCI #3 (rev 03)
00:1d.3 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) USB UHCI #4 (rev 03)
00:1d.7 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) USB2 EHCI Controller (rev 03)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev d3)
00:1f.0 ISA bridge: Intel Corporation 82801FB/FR (ICH6/ICH6R) LPC
Interface Bridge (rev 03)
00:1f.1 IDE interface: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
Family) IDE Controller (rev 03)
00:1f.2 IDE interface: Intel Corporation 82801FB/FW (ICH6/ICH6W) SATA
Controller (rev 03)
00:1f.3 SMBus: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family)
SMBus Controller (rev 03)
06:08.0 Ethernet controller: Intel Corporation 82562ET/EZ/GT/GZ - PRO/
100 VE (LOM) Ethernet Controller (rev 01)
diego0 ~ #
```

Para este caso nos interesa las líneas **00:1f.1** y **00:1f.2**. Activaremos las siguientes opciones en el kernel

```
▶▶ Device Drivers
    ▶▶ <*> Serial ATA (prod) and Parallel ATA (experimental) drivers
        [*]   ATA SFF support
        <*>     Intel ESB, ICH, PIIX3, PIIX4 PATA/SATA support
```

- **Dispositivos SCSI**

Tal vez no se disponga de un dispositivo de este tipo, pero estos controladores son tan genéricos que los usan también dispositivos de almacenamiento masivo USB y discos SATA.

```
▶▶ Device Drivers
    ▶▶ SCSI device support
        [*] legacy /proc/scsi/ support
        <*> SCSI disk support
        <*> SCSI CDROM support
        <*> SCSI generic support
```

- **Dispositivos de red**

Se reutilizará la información brindada por el comando *lspci* para activar los controladores de la placa de red; interesa la última línea.

```
▶▶ Device Drivers
    ▶▶ [*] Network device support
        ▶▶ [*]   Ethernet (10 or 100Mbit)
             [*]   EISA, VLB, PCI and on board controllers
             <*>   Intel(R) PRO/100+ support
```

Si está usando PPPoE para conectarse a Internet o está usando un módem dial-up, necesitará las siguientes opciones en el kernel

```
▶▶ Device Drivers
    ▶▶ [*] Network device support
        <*>   PPP (point-to-point protocol) support
        <*>     PPP support for async serial ports
        <*>     PPP support for sync tty ports
```

- **Dispositivos USB**

En general (reutilizando la salida de *lspci*)

```
▶▶ Device Drivers
    ▶▶ [*] USB support
        {*}   Support for Host-side USB
        [*]     USB device filesystem
        <*>     EHCI HCD (USB 2.0) support
        <*>     OHCI HCD support
        <M>     UHCI HCD (most Intel and VIA) support
        <M>   USB Modem (CDC ACM) support
        <M>   USB Printer support
        <*>   USB Mass Storage support
        [*]   The shared table of common (or usual) storage devices
```

Para dispositivos de entrada USB (como un ratón o teclado)

```
▶▶ Device Drivers
    ▶▶ [*] HID Devices
        <*>   USB Human Interface Device (full HID) support
```

- **Sistemas de archivos**

Seleccione *dentro* del kernel (es decir, no como módulos) los soportes para los sistemas de archivos que el kernel debe poder manejar. También seleccione *Virtual memory* y */proc file system*.

```
▶▶ File systems
    <*> Second extended fs support
    <*> Ext3 journalling file system support
    ▶▶ Pseudo filesystems
        [*] /proc file system support
```

```
[*] Virtual memory file system support (former shm fs)
```

Finalizar la configuración

Una vez que termine de configurar el kernel puede salir del programa y guardar los cambios. La configuración se escribe a un archivo llamado *.config* en el directorio en que ejecutó *make menuconfig* y se recomienda no editarlo a mano a menos que se tenga mucha experiencia y realmente se sepa lo que se hace.

Tal vez se quiera disponer de la configuración que acaba de hacer, pues se piensa en tunear posteriormente el kernel para darle un mejor rendimiento y se quiere tener un respaldo de la configuración que se acaba de hacer para poder volver atrás. En ese caso se puede: antes de salir del programa seleccionar *Save an Alternate Configuration File* para guardar la configuración actual en otro archivo diferente de *.config*; o también se puede, después de salir y guardar los cambios, copiar el archivo *.config* a otro

```
diego0 linux-2.6.29 # cp .config .config.krn0
diego0 linux-2.6.29 # _
```

y así se tendrá muchas configuraciones de respaldo, una cada vez que haga cierta cantidad de cambios.

6.5.2 . Compilación e instalación

La primera vez que se compila el kernel puede demorar mucho tiempo, pero las siguientes veces que se lo hace es más rápido porque sólo compilará lo que no se ha compilado antes.

Para compilar el kernel se puede ejecutar

```
diego0 linux-2.6.29 # make bzImage
     .
     .
     .
diego0 linux-2.6.29 # _
```

Ahora hay que copiar la imagen del kernel al directorio **/boot/**

```
diego0 linux-2.6.29 # find . -iname bzImage
./arch/x86_64/boot/bzImage
./arch/x86/boot/bzImage
diego0 linux-2.6.29 # l ./arch/x86_64/boot/bzImage
./arch/x86/boot/bzImage
lrwxrwxrwx 1 root root   22 2009-04-08 12:06
./arch/x86_64/boot/bzImage -> ../../x86/boot/bzImage
-rw-r--r-- 1 root root 2,3M 2009-04-08 12:06 ./arch/x86/boot/bzImage
diego0 linux-2.6.29 # cp arch/x86/boot/bzImage /boot/vmlinuz-2.6.29-
krn0
diego0 linux-2.6.29 # cp System.map /boot/System.map-2.6.29-krn0
```

```
diego0 linux-2.6.29 # _
```

Hasta ahora sólo se ha compilado el kernel, a continuación hay que compilar e instalar los módulos pues éstos son independientes del kernel.

```
diego0 linux-2.6.29 # make modules && make modules_install
     .
     .
     .
diego0 linux-2.6.29 # _
```

Hay que recordar que generalmente los módulos sólo se cargan automáticamente si así lo establece en el archivo específico que provea la distribución, y para saber qué módulos están disponibles podemos consultar el directorio **/lib/modules/** de la siguiente forma

```
diego0 ~ # find /lib/modules -iname "*.o" -or -iname "*.ko"
/lib/modules/2.6.24-23-generic/kernel/sound/ac97_bus.ko
/lib/modules/2.6.24-23-generic/kernel/sound/soundcore.ko
/lib/modules/2.6.24-23-generic/kernel/lib/libcrc32c.ko
/lib/modules/2.6.24-23-generic/kernel/lib/crc16.ko
/lib/modules/2.6.24-23-generic/kernel/lib/crc7.ko
/lib/modules/2.6.24-23-generic/kernel/lib/zlib_deflate/zlib_deflate.ko
/lib/modules/2.6.24-23-generic/kernel/lib/ts_kmp.ko
/lib/modules/2.6.24-23-generic/kernel/lib/reed_solomon/reed_solomon.ko
     .
     .
     .
diego0 ~ # _
```

Correr el nuevo kernel

Para correr el nuevo kernel habrá que reiniciar la computadora, pero antes habrá que configurar el gestor de arranque para que lo cargue. Esto se puede ver en “7 . Gestor de arranque GRUB”.

7 . Gestor de arranque GRUB

7.1 . Introducción

Un gestor de arranque, gestor multiarranque, cargador de arranque o *bootloader* es un programa relativamente sencillo diseñado para preparar el sistema para la ejecución de un sistema operativo, y también se encarga de levantarlo. Técnicamente, un gestor de arranque es un programa que puede cargar cualquier archivo ejecutable y que contiene un archivo de cabecera multiarranque en los primeros 8KB del archivo.

GNU GRUB (GRand Unified Bootloader) es un gestor de arranque múltiple que se usa para cargar uno de dos o más SSOO instalados en un mismo equipo. Una de sus características más importantes es que no es necesario crear una partición nueva o un kernel nuevo pues es posible cambiar todos los parámetros de arranque desde la consola de GRUB. También es capaz de examinar sistemas de archivos.

7.2 . Secuencia de inicio

- El BIOS busca un dispositivo de inicio (como un disco duro) y pasa el control al Registro Maestro de Buteo (Master Boot Record o MBR, los primeros 512B del disco duro).
- El MBR contiene la fase 1 de GRUB. Como el MBR es pequeño, la fase 1 carga solamente la siguiente fase del GRUB (ubicado físicamente en cualquier parte del disco duro), que puede ser la fase 1.5 o la 2.
- GRUB fase 1.5 está ubicada en los siguientes 30KB del disco duro. La fase 1.5 carga la fase 2.
- GRUB fase 2 (cargada por las fases 1 o 1.5) recibe el control, y presenta al usuario el menú de inicio de GRUB.
- GRUB carga el núcleo seleccionado por el usuario en la memoria y le pasa el control.

7.3 . Denominación de dispositivos de disco

La forma en que GRUB entiende los discos y las particiones es diferente a la que se viene utilizando. Para referirse a una partición de un disco de cualquier tipo (PATA, SATA, SCSI) se escribe "(hd**D**,**P**)", donde "D" se refiere al disco y "P"a la partición *y son números*. El primer disco es 0 (cero), el segundo 1, etc.; la primera partición de cualquier disco es 0 (cero), la segunda 1, y así sucesivamente.

Por ejemplo, si queremos referirnos a /dev/sdb4, estamos hablando del segundo disco cuarta partición; en vez de "b" escribimos "1" y en vez de "4" escribimos "3" resultando "(hd1,3)".

GRUB hace las designaciones de los dispositivos de acuerdo al BIOS.

Al cambiar la configuración del BIOS, cambian también las designaciones de los dispositivos.

Por ejemplo, si se cambia el orden de los dispositivos para el arranque, puede que haya que cambiar la configuración de GRUB también.

7.4 . Instalación

Una vez instalado el programa en sí por el medio que fuera (usando la utilidad de instalación de paquetes de la distribución o manualmente), se procede a la instalación de GRUB en el MBR y a la edición del archivo configuración. Se asume que estará instalado en /boot/grub/.

```
diego0 ~ # vi /boot/grub/grub.conf
     .
     .
     .
#### OPCIONES
## Dentro del listado, cuál es la opción que será ejecutada
## por defecto. La primera opción es cero. En vez de un
## número puede escribirse "saved"
default saved
## Tiempo de espera antes de ejecutar la opción por defecto
timeout 10

#### Listado de sistemas que pueden ejecutarse

title Gentoo Linux 2.6.24-r5
## Partición donde se encuentra se encuentra la imagen
## del kernel o el sistema operativo
root (hd0,4)
## Ruta hacia el kernel DESDE la raíz de la partición (es
## decir, relativo a la partición y no a la raíz del sistema);
## y parámetros pasados al kernel.
kernel /krn0 root=/dev/sda6

title Ubuntu 8.04.2, kernel 2.6.24-23-generic
root (hd0,8)
kernel /vmlinuz-2.6.24-23-generic root=UUID=02af89a0-1e6b-483f-86f0-
f49f86749753 ro quiet splash
## Opcional: imagen del Disco RAM de Inicio (init RAM Disk)
## alternativa
initrd /initrd.img-2.6.24-23-generic
savedefault

title ESTE ES UN SEPARADOR. Otros SSOO disponibles
root

## En el caso en que haya un SO Windows (ejemplo)
title Microsoft Windows XP Professional
root (hd0,0)
## Esta línea se explicará luego
```

```
savedefault
makeactive
chainloader +1
     .
     .    SALIR GUARDANDO LOS CAMBIOS
     .
diego0 ~ # _
```

Aclaraciones

- Si la opción "default" está seteada a "saved" entonces GRUB asumirá que la opción por defecto es la última que se ejecutó y que tenía la opción "savedefault". En el caso del ejemplo, la opción por defecto puede ser Ubuntu o Windows XP. El sistema (cualquiera de estos dos) que haya sido ejecutado por última vez será el que se ejecute por defecto; y, como Gentoo no tiene la opción "savedefault", no será ejecutado por defecto nunca.
- El Disco RAM de Inicio es un sistema de archivos usado por el kernel de Linux en forma temporal durante el buteo, generalmente para preparar el sistema antes de montar el directorio raíz.
- Si, por ejemplo, se utilizara otro esquema de particionamiento en el cual el directorio /boot/ no fuera una partición, sino que está dentro de la misma partición del directorio raíz, entonces el nombre de la imagen del kernel deberá comenzar obligatoriamente con/boot/. Para salvar estas diferencias, muchas distribuciones tienen por defecto un enlace simbólico a / boot/ en el mismo directorio (un bucle simbólico en el árbol) y en la configuración de GRUB escriben siempre "/boot/<kernel>" y es válido para cualquier caso.

Parámetros del kernel

Algunas distribuciones parchan el kernel de tal forma que reciben más (u otras) opciones, pero en general las siguientes serán válidas. Puede encontrarse una excelente lista detallada de muchos parámetros que puede recibir el kernel de Linux en **http://www.mjmwired.net/kernel/Documentation/kernel-parameters.txt**. En general, los parámetros que se le pasa al kernel *son independientes del gestor de arranque*.

- "root=<dispositivo_raíz>": indica el dispositivo de bloque en el cual se encuentra el directorio raíz. Nótese que puede sonar engañoso que se escriba "/dev/sda9" (por ejemplo), pues se está indicando el nombre del dispositivo de bloque que contienen a la raíz, relativamente a la raíz misma (el dispositivo "sda9" se encuentra en el directorio "dev/", que a su vez se encuentra en el directorio "/").
- "init=<programa>": correr el programa especificado en vez de **/sbin/init** como proceso init.
- "noinitrd": no cargar ningún Disco RAM de Inicio configurado.
- "nohalt": incrementa el consumo de energía como consecuencia de una mejora en el desempeño en ciertos aspectos, y puede ser beneficioso para servidores de red o sistemas en tiempo real.
- "vga=<código>": selecciona un modo de video particular. En realidad, es un parámetro para el cargador de arranque (válido para GRUB y LILO).

	640x480	**800x600**	**1024x768**	**1280x1024**
256	0x301	0x303	0x305	0x307
32k	0x310	0x313	0x316	0x319
64k	0x311	0x314	0x317	0x31A
16M	0x312	0x315	0x318	0x31B

Tabla 7: Valores aceptables para la opción "vga="

Instalación en el MBR

Esta tarea puede llevarse a cabo con la herramienta *grub-install*; bastaría leer el breve pero conciso *man grub-install*. Por motivos profesionales se verá la forma manual instructiva, tal vez más poderosa.

Se entrará al shell de GRUB, ejecutando *grub*. Se puede especificar el parámetro *--nofloppy* si es que no tenemos una disketera y queremos evitar la comprobación de la que no existe.

```
diego0 ~ # grub --nofloppy
     .
     .
     .
    GNU GRUB  version 0.97  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported. For the first word,
TAB lists possible command completions. Anywhere else TAB lists the
possible completions of a device/filename. ]

grub> _
```

El sistema de *terminación de palabras* de GRUB debería facilitar las tareas que se realizan (“completion”, en español sería algo así como “completación” o “compleción”, si tales palabras existieran, y se refiere a un mecanismo de ayuda para terminar la palabra, orden o *token* que se está escribiendo).

Sólo restan dos pasos sencillos: especificar dónde se encuentra la partición **/boot/** del sistema e instalar GRUB en el MBR del disco. En el ejemplo que sigue se ejecutarán dos comandos que harán las acciones respectivas, indicando que en **/dev/sda9** está la partición /boot/ y que el disco en el cual queremos instalar GRUB es **/dev/sda**.

```
grub> root (hd0,8)
grub> setup (hd0)
grub> quit
     .
     .
     .
diego0 ~ # _
```

El significado del tercer comando es transparente.

Compartiendo el disco con otro SO

En el caso que se comparta el disco rígido con otro SO (para el caso MS Windows XP) y éste haya sido instalado después que Linux habrá que repetir este último paso (la instalación en el MBR), pues este SO no dispone de un gestor multiarranque que permita arrancar Linux (al menos no convencionalmente).

Una vez instalado MS Windows XP lisa y llanamente no se podrá ejecutar el Linux que se había instalado en ese disco, pues no se dispone del medio para levantarlo durante el inicio de la máquina. Lo que habrá que hacer es arrancar la máquina con un LiveCD o DVD de alguna distribución (lo más común) y ejecutar los comandos ya vistos, dentro del shell de GRUB. Hecho esto, para ser capaces de arrancar el SO Windows sólo habrá que agregar las correspondientes líneas al archivo de configuración de GRUB, como ya se ha visto.

8 . Administración de usuarios

8.1 . Introducción

Linux es un SO multiusuario y delega mucha importancia en ésto, y la administración de usuarios se convierte en una importante tarea de seguridad que un administrador debe llevar a cabo en forma cabal. En "4.10 . Sistema de permisos" ya se describió en forma básica el funcionamiento de los usuarios y grupos en Linux.

Archivos de configuración personal

Como ya se sabe en el directorio /etc/ se encuentra muchos archivos que configuran el comportamiento predeterminado del sistema y de las aplicaciones de usuario y que son suceptibles de ser modificados para adaptarse más a los requerimientos de ese sistema en particular. Pero también sabemos que Linux es un sistema multiusuario, lo que quiere decir que muchos usuarios comparten el mismo equipo y cada uno puede (y seguramente es así) necesitar una configuración particular de las aplicaciones de usuario.

La solución implementada es la existencia de archivos de configuración en el directorio /etc/ que controlan todos los aspectos de la aplicación que son configurables, y la adición de archivos ocultos en los directorios personales de cada usuario para complementar una configuración aún más personal. Estos archivos ocultos generalmente tienen el mismo nombre que sus pares en /etc/ (a diferencia, claro, por el punto que los precede).

Por ejemplo, podemos encontrar los archivos */etc/vim/vimrc* y */root/.vimrc*.

8.2 . Archivos relacionados

8.2.1 . /etc/passwd

En este archivo se guarda la información que indica qué usuarios existen y otros datos sobre los usuarios. Cada usuario está representado con una línea de siete campos cada una, cada campo separado por dos puntos (":"). Es posible crear manualmente usuarios editando directamente este archivo, aunque no es recomendado.

Nombre	Nombre del usuario en el sistema. Se recomienda que no tenga letras mayúsculas.
Contraseña	Un asterisco ("*"), un letra equis minúscula ("x") o la contraseña del usuario encriptada usando el algoritmo de preferencia del sistema.
UID	User ID. Número que identifica al usuario. Dos o más usuarios pueden tener un mismo UID, el sistema los tratará como un mismo usuario.
GID	Group ID. Número que identifica al grupo principal del usuario.

GECOS	General Electrics Comprehensive Operating System. Opcional. Tiene fines informativos solamente. Generalmente tiene el nombre completo del usuario.
home	Directorio personal del usuario.
Shell	Programa que se ejecuta al loguearse (por defecto **/bin/sh**). Si el ejecutable no existe el usuario no podrá loguearse con **login**.

Tabla 8: Campos de cada línea del archivo /etc/passwd

```
diego0 ~ # ls -lah --color=always /etc/passwd
-rw-r--r-- 1 root root 1,2K dic 29 10:36 /etc/passwd
diego0 ~ # _
```

Como se puede ver, este archivo puede ser leído por cualquier usuario, pero escrito sólo por el superusuario (que lo posee). Esto puede resultar en una falencia de seguridad pues podría leerse este archivo, leer la contraseña encriptada de *root* y armados de paciencia probar hasta dar con el valor de la contraseña. Por esto es una buena práctica tener contraseñas complicadas y cambiarlas periódicamente.

Por otro lado, la manera más segura de guardar las contraseñas es haciendo uso del archivo **/etc/shadow**, que se explica adelante. Este archivo sólo tiene los permisos de lecto-escritura para el superusuario, y ninguno para otro usuario, lo que lo hace más seguro. Si en la columna de contraseña aparece una "x" entonces las contraseñas están guardadas en este archivo (de seguro este será el caso siempre, el viejo sistema de contraseñas encriptadas dentro de /etc/passwd está prácticamente erradicado).

Ejemplo

```
diego0 ~ # cat /etc/passwd | line
root:x:0:0:root:/root:/bin/bash
diego0 ~ # _
```

8.2.2 . /etc/shadow

Este archivo tiene una sintaxis parecida a la del archivo anterior, lo que difiere son los campos.

Nombre	Nombre de logueo del usuario
Contraseña	Contraseña encriptada. Consiste en 13 a 24 caracteres de un alfabeto de 64 caracteres que pueden ser de "a" hasta "z" (anglosajonas, sin "ñ") y capitalizados, de cero a nueve, "\." y "/". Si este campo contiene alguna cadena que no es un valor de contraseña válido (como "!" o "*") el usuario no podrá usar una contraseña UNIX para

	loguearse.
Último cambio	Días transcurridos desde 01/01/1970 hasta el último cambio de la contraseña.
Es modificable	Días que deben pasar para que la contraseña sea modificable
Debe modificarse	Días que deben pasar para que la contraseña deba ser cambiada.
Aviso	Días antes de la expiración de la contraseña en la que el usuario es advertido de esto.
Desactivación	Días que deben pasar antes de la desactivación de la cuenta tras expirar la contraseña.
Caduca	Fecha en que caduca la cuenta. Se cuenta desde el 01/01/1970.
Reservado	Un campo reservado.

Tabla 9: Campos de cada línea del archivo /etc/shadow

Ejemplo

```
diego0 ~ # cat /etc/shadow | line
root:$1$/o9wdDua$ImfoQyjv31lU49tm/CYKb1:14036:0:::::
diego0 ~ # _
```

8.2.3 . /etc/group

Describe los grupos a los que pertenecen los usuarios.

Nombre	Nombre del grupo.
Contraseña	Contrseña del grupo. Si este campo está vacío no se necesita contraseña.
GID	Group ID. La identificación numérica del grupo
Usuarios	Lista de todos los usuarios del grupo, separado por comas.

Tabla 10: Campos de cada línea del archivo /etc/group

Ejemplo

```
diego0 ~ # cat /etc/group | grep root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
disk::6:root,adm,haldaemon
```

```
wheel::10:root,diego
floppy::11:root,haldaemon,diego
dialout::20:root
tape::26:root
video::27:root
diego0 ~ # _
```

8.2.4 . /etc/login.defs

Controla aspectos importantes en la administración de usuarios. Una descripción considerable de este archivo se puede ver en su págna *man*, aquí sólo veremos algunas configuraciones básicas. A las variables buleanas se les puede asignar “yes” o “no” (por defecto).

- FAIL_DELAY: demora en segundos antes de poder intentar loguearse nuevamente luego de ingresar una contraseña inválida.
- LOGIN_RETRIES: cantidad de reintentos en caso de ingresar una contraseña mala.
- LOGIN_TIMEOUT: máximo tiempo para loguearse (en segundos).
- PASS_MAX_DAYS: la mayor cantidad de días que se puede usar una contraseña.
- PASS_MIN_DAYS: la mínima cantidad de días que una contraseña se pude usar.

8.3 . Comandos relacionados

groupadd

Crea una nueva cuenta de grupo usando los valores pasados por argumentos más los valores por defecto del sistema. El nuevo grupo será ingresado en los respectivos archivos según corresponda. Opciones

- -f, --force: provoca que el comando salga con un estado de salidad de éxito si el grupo ya existe. Si se usa con la opción -g y el GID ya existe entonces se elige otro GID.
- -g, --gid GID: valor numérico del GID. Por defecto se usa el menor ID mayor que 999 y mayor que cualquier otro grupo. Valores entre cero y 999 típicamente son para cuentas del sistema.
- -K, --key CLAVE=VALOR: sobreescribe el valor VALOR para la clave CLAVE en /etc/login.defs. Pueden especificarse múltiples opciones de este tipo.
- -o, --non-unique: permite agregar un grupo con un GID no único.
- -r, --system: crea un grupo del sistema.

groupdel

Elimina una cuenta de grupo al borrar todas las entradas de ese grupo en los archivos de configuración del sistema. El grupo que se especifique debe existir. El administrador deberá checkear manualmente todos los sistemas de archivos para asegurarse de que no quedan archivos con este grupo como grupo propietario.

groupmod

Modifica la definición de un grupo en el sistema. Opciones:

- -g, --gid NUEVO_GID: el GID del grupo se cambiará a NUEVO_GID. Los archivos que tienen el viejo GID y que deben tener continuar perteneciendo a este grupo deben ser modificados manualmente.
- -n, --new-name NUEVO_GRUPO: el nombre del grupo se cambiará a NUEVO_GRUPO.
- *-o, --non-unique*: cuando se está usando la opción *-g* permite cambiar el GID a un alor no único.

passwd

Cambia contraseñas. Si se ejecuta sin argumentos cambia la contraseña del usuario que lo invoca. Opciones:

- -o, --force: Desactiva las verificaciones de validación sobre la nueva contraseña. Sólo puede ser utilizada por el superusuario, y está pensado para permitirle asignar contraseñas iniciales simples.
- -s, --shell: cambia el shell del usuario mediante la invocación del programa /usr/bin/chsh con los parámetros recibidos que no son opciones.
- -q, --quiet, --silent: no se indica que la contraseña se haya cambiado.

su

Ejecuta una shell con GID y UID distintos. Opciones:

- -c COMANDO, --command=COMANDO: pasa COMANDO, una única línea a ejecutar, a la shell con la opción -c en vez de ejecutar una shell interactiva.
- -, -l, --login: provoca que la shell a ejecutar sea una shell de login.
- *-s, --shell SHELL*: ejecuta el SHELL en vez de el shell del USUARIO especificada en el archivo /etc/passwd, a menos que el usuario que ejecute su no sea el superusuario y el shell del USUARIO esté restringido.

useradd

Crea un nuevo usuario o actualiza la información de un usuario que ya existe. Cuando se invoca con la opción *-D* crea una nueva cuenta de usuario usando los valores ingresados más los valores por defecto del sistema. Dependiendo de las opciones que se indiquen se pueden actualizar los archivos del sistema y también crear el directorio personal del nuevo usuario y también copiar los archivos iniciales. Opciones:

- -c, --comment COMENTARIO: algún comentario.
- -d, --home HOME_DIR: usar HOME_DIR como directorio personal del usuario.
- -g, --gid GRUPO: el nombre o número del grupo de logueo del usuario. Debe referirse a un grupo que ya existe. El grupo por defecto es 1 (uno) o el valor especificado en /etc/default/useradd.
- -G, --groups: lista separada por comas de grupos suplementarios al que el usuario pertenecerá.
- -k, --skel SKEL_DIR: el directorio skeleton, cuyo contenido será copiado al directorio personal del usuario cuando sea creado por useradd. Esta opción es válida sólo con la opción -m. Si esta opción no se setea, se usará el directorio skeleton definido en /etc/default/useradd o, por defecto, /etc/skel/.

- -K, --key CLAVE=VALOR: sobreescribe el valor VALOR para la clave CLAVE en /etc/login.defs. Pueden especificarse múltiples opciones de este tipo.
- -m, --create-home: crear un directorio personal si es que no existe. Los archivos y directorios contenidos en el directorio skeleton se copiarán a este directorio.
- -N, --no-user-group: no crear un grupo con el mismo nombre que el usuario, pero agregar el usuario al grupo especificado con la opción -g o al grupo especificado en la variable GROUP de /etc/default/useradd.
- -o, --non-unique: permitir la creación de un usuario con un UID no único.
- -r, --system: crear una cuenta del sistema.
- -s, --shell SHELL: el nombre del shell de logueo del usuario.
- -u, --uid UID: el valor numérico del UID. Por defecto se usa el menor ID mayor que 999 y mayor que cualquier otro grupo. Valores entre cero y 999 típicamente son para cuentas del sistema.
- -U, --user-group: crear un grupo con el mismo nombre del usuario y agregar este usuario al grupo.

userdel

Modifica los archivos de cuentas de usuarios del sistema al eliminar todas las entradas que se refieren al *login* del nombre de usuario. El usuario debe existir. Opciones:

- -f, --force: fuerza la eliminación del usuario incluso si está logueado. También fuerza el eliminado del directorio personal del usuario y de su cola de correo, sin importar si comparte con otro usuario el directorio personal o si la cola de correo no la posee el usuario especificado. Si USERGROUPS_ENAB vale “yes” en /etc/login.defs y si existe un grupo con el mismo nombre que el usuario que se borra entonces será borrado el grupo también, incluso si es el grupo principal de algún otro usuario. Esta opción es radical y peligrosa y puede dejar el sistema en un estado inestable.
- -r, --remove: se eliminará tanto el directorio personal del usuario como su contenido y la cola de correo (“mail spool”, definido por la variable MAIL_DIR en /etc/login.defs). Los archivos que se encuentran en otros lugares deberán ser eliminados manualmente.

usermod

Modifica los archivos de cuentas de usuarios del sistema para reflejar los cambios que se indican por línea de comandos. Opciones:

- -d, --home HOME_DIR: el nuevo directorio personal del usuario.
- -g, --gid GRUPO: el nombre o GID del grupo de logueo del usuario.
- -G, --groups GRUPO0,GRUPO1,...,GRUPOn: lista suplementaria separada por comas de grupos a los que también pertenece el usuario.
- -l, --login NUEVO_LOGIN: sólo se cambia el nombre del usuario a NUEVO_LOGIN. Podría cambiarse manualmente para reflejar el cambio.
- -o, --non-unique: permite cambiar el UID a un valor no único.
- -s, --shell SHELL: modifica el shell de logueo del usuario.
- -u, --uid UID: cambiar el UID del usuario. Se modificarán los archivos contenidos en el directorio personal del usuario para que se ajusten a este cambio, pero los archivos fuera de este directorio deberán ser alterados manualmente.

9 . Comandos de red

9.1 . Definiciones generales

MAC Address (“Media Access Control Address”, o “Dirección de Control de Acceso al Medio”) o BIA (“Burned In Address”, “Dirección quemada dentro”)

Es un identificador de 6B (48b) que generalmente se representa como un sexteto ordenado de dos dígitos hexadecimales separados por “dos puntos” (“:”), y sirve para identificar en forma única (a nivel mundial) una placa de red. Es decir, cada placa de red tiene una única MAC Address.

Protocolo de Internet, Internet Protocol o IP

Es un conjunto de estándares tecnológicos y especificaciones técnicas para la transmisión de datos a través de una red de paquetes conmutados. La información se organiza en bloques de datos conocidos como *paquetes* o *datagramas*.

Dirección IP

Es un número de 32bits separado en cuartetos de 8bits cuyo propósito es la identificación lógica y jerárquica de las interfaces de los dispositivos (generalmente una computadora) dentro de una red que utiliza el *Protocolo de Internet* (IP). En base diez se lo representa como un cuarteto de números (cada uno va desde cero a 255) separados por puntos. También se habla de “primer octeto”, “segundo octeto”, etc. refiriéndose a los respectivos octetos de bits.

Host o anfitrión

Puede referirse a un servidor de una red (servidor de algún servicio, valga la redundancia) o a cualquier equipo identificable conectado a una red.

Máscara de subred o máscara de red

La máscara de subred indica el rango de máquinas en la red, y permite clasificar las redes por lo menos en tres clases. Específicamente, la máscara de red indica la cantidad de bits que corresponden a la identificación de la red.

Clase de red	Octetos de bits de la IP para identificar la red	Octetos de bits de la IP para identificar el host	Máscara de subred
Clase A	El primero	Los tres últimos	255.0.0.0
Clase B	Los dos primeros	Los dos últimos	255.255.0.0
Clase C	Los tres primeros	El último	255.255.255.0

Tabla 11: Clases de IP

Se suele representar a una máquina con su IP seguida de una barra de división y su máscara de red. La forma mostrada en la tabla corresponde a la forma larga. La forma corta consiste sólo

de un número que indica la cantidad de bits usados para la red.

Dos máquinas pueden tener al mismo tiempo una misma IP y estar conectadas físiscamente, pero entonces deben pertenecer a redes diferentes. Por ejemplo, si la IP de estas máquinas es 172.16.0.4, entonces puede que una sea 172.16.0.4/16 y la otra 172.16.0.4/24 (clase B y C respectivamente).

Dirección Broadcast (o dirección de difusión)

Supongamos que en una red se tienen al menos dos máquinas conectadas por un switch en una red de clase B. En principio, el cable de red de cada una las conecta sólo con el switch, por lo que antes de conectarse unas con otras deben pasar por ese lugar al que concurren las demás, y de alguna forma identificarse e identificar la computadora a la que se quiere conectar.

El proceso de conexión de dos máquinas es un poco más extenso pero más o menos es el siguiente

- La máquina de IP 172.16.0.4 (la "4" en adelante) quiere conectarse a la máquina de IP 172.16.0.5 (la "5" en adelante).
- La "4" manda un paquete llamado *de broadcast* al switch, que envía ese paquete a todas las máquinas de la red. Este paquete contiene información que indica que la "4/16" se quiere conectar con la "5/16".
- Sólo la máquina configurada con la IP "5/16" responde el paquete de broadcast directamente a la "4/16" a través del switch. La "5" puede responder directamente a la "4" porque el paquete de broadcast contenía la información necesaria para que la "5" pudiera efectiva y *directamente* conectarse con la "4".
- La "4" recibe ese paquete de respuesta, que contiene la información para que la pueda ubicar a la "5". Otros paquetes que sirven para el control y la sincronización son enviados entre las máquinas, y luego (finalmente) comienza la comunicación efectiva de la información entre las dos máquinas.

El broadcast es una dirección IP especial, y, como puede verse, el broadcast sólo sirve para ayudar a que dos máquinas se "encuentren" físicamente. Esta IP comienza con los bits de la red y termina con todos los restantes bits en 1 (uno), por lo que en general siempre veremos que terminan en 255.

Pero la forma correcta de calcular una dirección de broadcast es usando la máscara de red: se copian de una IP todos los bits correspondientes a la red (que los indica la máscara) y el resto se los pone en 1. De esta forma, la dirección de broadcast del ejemplo anterior (es decir, la dirección IP a la que la "4" envió el primer paquete) es 172.16.255.255.

Dirección Loopback

Esta dirección es un dispositivo de red virtual que hace referencia al host mismo, sin importar los parámetros de configuración de red que tenga. Se utiliza como herramienta de diagnóstico y para otras tareas. En cualquier equipo, esta dirección es 127.0.0.1.

IP válidas

Las IP válidas para los hosts son todas aquellas cuyos ***primeros y últimos***

octetos ***no son cero ni 255***, pues ésas son IP reservadas.

Ninguna IP válida tiene su primer octeto en cero.

Las IP cuyo último octeto es cero identifican ***redes***. Las IP que comienzan o terminan con 255 son direcciones de Broadcast.

Gateway, Puerta de enlace o Pasarela

Es un dispositivo que sirve como conexión entre dos redes y/o como traductor de protocolos. Generalmente se trata de una computadora a la que se conectan en LAN varias otras, y tienen acceso a una red más amplia (como Internet) o diferente a través de ésta.

Dispositivos de red en Linux

Se asumirá que se usa una placa de red ethernet. No se tratará el tema de placas de red FDDI (fibra óptica) por ser no tan común y porque sale del marco general del manual.

El nombramiento de las placas (el orden que tienen: si una placa es eth1 o si es eth2) depende del sistema que tengamos.

En sistemas antiguos la manipulación de dispositivos la hacía *devfs*, y se nombraban los dispositivos a medida que se cargaban los módulos que los controlaban; si había dos o más dispositivos idénticos se respetaba entonces el orden físico de conexión (la numeración de los puertos en que se conectaban los dispositivos).

Actualmente se utiliza el sistema *udev*, que "recuerda" la MAC Address. La primera placa ethernet que se detecta la llama "eth0". Si se desconecta esta placa y se conecta otra, esta otra se llamará "eth1" y así sucesivamente. Si eventualmente se reconectara la primera placa, entonces se la reconocería por su MAC Address y nuevamente se la llamaría "eth0".

9.2 . Operaciones básicas

9.2.1 . Probar una conexión

Para probar una conexión se puede usar el comando *ping*, que envía paquetes a una dirección (o dominio) y espera una respuesta. Algunas de las opciones que puede recibir son:

- *-b*: permite hacer ping a una dirección broadcast.
- *-c CANT*: se detiene luego de enviar y recibir CANT paquetes.
- *-f*: envío masivo de pings. Envía paquetes a la misma velocidad a la que regresan o cien veces por segundo, lo que sea mayor. Para facilitar una visión general instantánea de la conexión, *ping* va escribiendo un punto (".") por cada paquete que envía; y por cada paquete que recibe borra un punto. De este modo si estamos probando una conexión en forma intensiva y la conexión se satura empezaremos a ver muchos puntos (paquetes enviados y sin respuesta).

- *-i INTERV*: intervalo de espera entre el envío de paquetes. Por defecto espera un segundo (ninguno con la opción -f).
- *-I {INTERFAZ | DIR_IP}*: establece la dirección remitente a la especificada. El argumento puede ser una IP numérica o el nombre de la interfaz.
- *-r*: Pasa por alto las tablas de encaminamiento y envía datos directamente a un ordenador en una red conectada a la propia. Si el ordenador receptor no está en una red con conexión directa, se devuelve un error.
- *-s*: Especifica el número de bytes de datos que se van a enviar. La cantidad por defecto es 56, que pasan a ser 64 bytes de datos ICMP cuando se combinan con los 8 bytes de los datos de la cabecera ICMP**.**

Ejemplo

Conexión exitosa entre el sitio www.google.com.ar y el equipo.

```
diego0 ~ # ping www.google.com.ar -c 5 | grep -nE "*"
1:PING www.l.google.com (64.233.163.103) 56(84) bytes of data.
2:64 bytes from bs-in-f103.google.com (64.233.163.103): icmp_seq=1
ttl=245 time=76.9 ms
3:64 bytes from bs-in-f103.google.com (64.233.163.103): icmp_seq=2
ttl=245 time=80.6 ms
4:64 bytes from bs-in-f103.google.com (64.233.163.103): icmp_seq=3
ttl=245 time=74.1 ms
5:64 bytes from bs-in-f103.google.com (64.233.163.103): icmp_seq=4
ttl=245 time=76.4 ms
6:64 bytes from bs-in-f103.google.com (64.233.163.103): icmp_seq=5
ttl=245 time=80.2 ms
7:
8:--- www.l.google.com ping statistics ---
9:5 packets transmitted, 5 received, 0% packet loss, time 3999ms
10:rtt min/avg/max/mdev = 74.148/77.682/80.693/2.480 ms
diego0 ~ # _
```

Se puede ver el tiempo de demora entre el envío y la recepción de un paquete.
El siguiente es un ejemplo de una conexión fallida

```
diego0 ~ # ping www.google.com.ar -c 5
ping: unknown host www.google.com.ar
diego0 ~ #
```

Otro ejemplo de conexión fallida

```
diego0 ~ # ping 172.16.0.99 -c 5
PING 172.16.0.99 (172.16.0.99) 56(84) bytes of data.
From 172.16.0.4 icmp_seq=2 Destination Host Unreachable
From 172.16.0.4 icmp_seq=3 Destination Host Unreachable
From 172.16.0.4 icmp_seq=4 Destination Host Unreachable
From 172.16.0.4 icmp_seq=5 Destination Host Unreachable
```

```
--- 172.16.0.99 ping statistics ---
5 packets transmitted, 0 received, +4 errors, 100% packet loss, time
3999ms, pipe 3
diego0 ~ # _
```

9.2.2 . Levantar una interfaz de red

Configuración automática

Actualmente la mayoría de las distribuciones tienen mucho éxito en la configuración automática de red. Esto se hace generalmente cuando se corre el medio de instalación sobre la máquina cuando se está por instalar Linux.

Existen comandos que configuran en forma automática la red. Una de ellas es *net-setup*. Esta herramienta muestra un entorno visual en modo texto y va haciendo preguntas sencillas sobre la red que se está configurando. Si no se le pasa como argumento el dispositivo que queremos configurar, entonces nos listará los dispositivos disponibles. Cuando se termine con el programa las funciones de red deberían estar habilitadas. Esta herramienta funciona en varias distribuciones como Gentoo y algunas derivadas (tal vez sea posible encontrarla en otras distribuciones).

Configuración manual

Antes de usar cualquier dispositivo físico en el sistema, hay que verificar que el/los módulos que lo controlan estén cargados. Para esto, puede usarse el siguiente comando

```
diego0 ~ # lsmod
Module                  Size  Used by
af_packet              27272  2
i915                   38144  2
drm                   105896  3 i915
ppdev                  11400  0
ipv6                  311848  14
cpufreq_ondemand       11152  0
cpufreq_stats           8416  0
freq_table              6464  2 cpufreq_ondemand,cpufreq_stats
cpufreq_powersave       3200  0
cpufreq_userspace       6180  0
cpufreq_conservative    10632  0
video                  23444  0
output                  5632  1 video
     .
     .
     .
diego0 ~ # _
```

Este comando lista los módulos del kernel que están cargados. En caso de que el módulo no lo

estuviese, habrá que cargarlo con modprobe. No hace falta decir que si el controlador de la placa de red fue compilado dentro del kernel entonces no hace falta cargarlo.

Para probar si la tarjea de red está funcionando se puede ejecutar el comando *ifconfig* con el nombre del dispositivo como argumento. Una tarjeta de red detectada y una no detectada (respectivamente) podrían verse como

```
diego0 ~ # ifconfig eth0
eth0      Link encap:Ethernet  direcciónHW 00:13:20:50:3a:4d
          inet dirección:172.16.0.27  Difusión:172.16.0.255
Máscara:255.255.0.0
          dirección inet6: fe80::213:20ff:fe50:3a4d/64 Alcance:Vínculo
          ARRIBA DIFUSIÓN CORRIENDO MULTICAST  MTU:1500  Métrica:1
          RX packets:2598 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2070 errors:0 dropped:0 overruns:0 carrier:0
          colisiones:0 txqueuelen:1000
          RX bytes:215865 (210.8 KB)  TX bytes:537545 (524.9 KB)
diego0 ~ # ifconfig eth1
eth1: error al obtener información sobre la interfaz: Dispositivo no
encontrado
diego0 ~ # _
```

Si se ejecuta *ifconfig* sin argumentos pueden verse todos los dispositivos de red detectados e información sobre su configuración

```
diego0 ~ # ifconfig
eth0      Link encap:Ethernet  direcciónHW 00:13:20:50:3a:4d
          inet dirección:172.16.0.27  Difusión:172.16.0.255
Máscara:255.255.0.0
          dirección inet6: fe80::213:20ff:fe50:3a4d/64 Alcance:Vínculo
          ARRIBA DIFUSIÓN CORRIENDO MULTICAST  MTU:1500  Métrica:1
          RX packets:2631 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2093 errors:0 dropped:0 overruns:0 carrier:0
          colisiones:0 txqueuelen:1000
          RX bytes:218619 (213.4 KB)  TX bytes:540727 (528.0 KB)

lo        Link encap:Bucle local
          inet dirección:127.0.0.1  Máscara:255.0.0.0
          dirección inet6: ::1/128 Alcance:Anfitrión
          ARRIBA LOOPBACK CORRIENDO  MTU:16436  Métrica:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          colisiones:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
diego0 ~ # _
```

Usar DHCP (“Dynamic Host Configuration Protocol” o “Protocolo de Configuración Dinámica de Anfitrión”)

DHCP es un protocolo de red para redes IP que permite a un host obtener sus parámetros de

configuración (como la dirección IP, la máscara de red, dirección broadcast, puerta de enlace, etc.) en forma automática. Sólo funciona si se tiene un servidor DHCP corriendo en la red o si el ISP ("Internet Service Provider" o "Proveedor de Servicio Inernet") lo provee.

La herramienta que se usa en este manual es el comando *dhcpcd* o *dhcpcd-bin* (dependiendo de la distribución). Recibe como parámetro el nombre del dispositivo que se configurará.

```
diego0 ~ # dhcpcd eth0
     .
     .
     .
diego0 ~ # _
```

Configuración directa

Para configurar en forma directa una tarjeta de red hay que tener conocimiento de los siguientes datos para la configuración:

I. IP de la interfaz.
II. Máscara de red de la interfaz.
III. Dirección broadcast de la interfaz.
IV. Si hubiera una, la puerta de enlace.
V. La IP del servidor de nombres de Internet.

Configuración directa con *ifconfig* y *route* y alternativamente con *ip*

Se recomienda seguir por separado los ejemplos hechos con ifconfig y route, y los realizados con ip.

I. *Parámetros de configuración de red de la interfaz*

Se ejecuta ifconfig para asignarle los siguientes parámetros a la interfaz

```
diego0 ~ # ifconfig eth0 172.16.0.30 netmask 255.255.0.0 broadcast
172.16.255.255 up
diego0 ~ # _
```

Alternativa con ip

```
diego0 ~ # ip a a 172.16.0.30/16 dev eth0
diego0 ~ # _
```

Para confirmar los cambios

```
diego0 ~ # ifconfig eth0
eth0      Link encap:Ethernet  direcciónHW 00:13:20:50:3a:4d
          inet dirección:172.16.0.30  Difusión:172.16.255.255
Máscara:255.255.0.0
          dirección inet6: fe80::213:20ff:fe50:3a4d/64 Alcance:Vínculo
          ARRIBA DIFUSIÓN CORRIENDO MULTICAST  MTU:1500  Métrica:1
          RX packets:990 errors:0 dropped:0 overruns:0 frame:0
```

```
          TX packets:633 errors:0 dropped:0 overruns:0 carrier:0
          colisiones:0 txqueuelen:1000
          RX bytes:92759 (90.5 KB)  TX bytes:79162 (77.3 KB)
diego0 ~ # _
```

Alternativamente con ip

```
diego0 ~ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
qlen 1000
    link/ether 00:13:20:50:3a:4d brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.30/16 brd 172.16.255.255 scope global eth0
    inet6 fe80::213:20ff:fe50:3a4d/64 scope link
       valid_lft forever preferred_lft forever
diego0 ~ # _
```

II. *Enrutamiento*

Se puede ver el enrutamiento con el siguiente comando

```
diego0 ~ # route -n | grep -nE "*"
1:Tabla de rutas IP del núcleo
2:Destino         Pasarela        Genmask         Indic Métric Ref
Uso Interfaz
3:172.16.0.0      *               255.255.0.0     U     0      0
0 eth0
diego0 ~ # _
```

Alternativamente con ip

```
diego0 ~ # ip ro
172.16.0.0/16 dev eth0  proto kernel  scope link  src 172.16.0.30
diego0 ~ # _
```

Si oportuno (como en este caso), se puede agregar una pasarela para enrutar por defecto

```
diego0 ~ # route add default gw 172.16.0.3
diego0 ~ # _
```

Alternativamente con ip

```
diego0 ~ # ip ro a via 172.16.0.3 dev eth0
diego0 ~ # _
```

Ahora se puede comprobar nuevamente las rutas

```
diego0 ~ # route -n | grep -nE "*"
1:Tabla de rutas IP del núcleo
2:Destino         Pasarela        Genmask         Indic Métric Ref
Uso Interfaz
3:172.16.0.0      0.0.0.0         255.255.0.0     U     0      0
0 eth0
4:0.0.0.0         172.16.0.3      0.0.0.0         UG    0      0
0 eth0
diego0 ~ # _
```

Alternativamente con ip

```
diego0 ~ # ip ro
172.16.0.0/16 dev eth0  proto kernel  scope link  src 172.16.0.30
default via 172.16.0.3 dev eth0
diego0 ~ # _
```

III. *Edición del archivo /etc/resolv.conf*

Este archivo configura el *resolver* (conjunto de rutinas que provee acceso al Sistema de Nombre de Dominios —DNS o "Domain Name System"— de Internet). Las opciones se escriben una por línea, primero el nombre de la opción, un espacio y luego el valor que se le asigna. Algunas de ellas pueden ser

- nameserver: la IP de un servidor que el resolver debería consultar. Por lo general sólo puede haber hasta tres, escritos en líneas independientes.
- domain: nombre del dominio local.
- search: lista de búsqueda para la búsqueda de nombre de host.

El siguiente es un ejemplo real de configuración de este archivo.

```
diego0 ~ # cat /etc/resolv.conf
search fortix.com.ar
nameserver 200.45.191.35
diego0 ~ # _
```

Éste es otro ejemplo

```
diego0 ~ # cat /etc/resolv.conf
# Generated by dhcpcd
# /etc/resolv.conf.head can replace this line
# /etc/resolv.conf.tail can replace this line
search fortix.com.ar
nameserver 200.45.191.35
nameserver 200.45.191.40
diego0 ~ # _
```

Configuración permanente (sobre archivos)

La configuración que se realizó con anterioridad es volátil: no sobrevive a un reinicio. Para hacerlo debemos asentar la información de configuración sobre archivos. Estos archivos varían entre distribuciones, así que no hay una generalidad razonable para asentar en forma estática esta configuración. Sí se puede, como ya se viene haciendo, establecer pequeñas generalidades para grandes grupos de distribuciones.

Configuración permanente (sobre archivos) para distribuciones basadas en Debian

Utilizan un archivo llamado */etc/network/interfaces* para la configuración estática. Los comandos *ifup* e *ifdown* levantan y bajan, respectivamente, una interfaz.

- Archivo */etc/network/interfaces*: las primeras dos líneas deberían ser las siguientes

```
auto lo
iface lo inet loopback
```

Las líneas que comienzan con "#" son ignoradas. Las líneas que comienzan con "auto" indican las interfaces que pueden levantarse o bajarse en forma automática.

La configuración de las interfaces es como sigue

```
iface <nombre_interfaz:[lo | eth0 | ...]> inet <flia_dirección_inet:
[loopback | static | dhcp | ppp]>
```

Luego de esta línea siguen las opciones de la interfaz, una por línea.

- Opciones para <flia_dirección_inet> = "loopback"
  - Sin opciones. Debería estar por defecto.
- Opciones para <flia_dirección_inet> = "static":
  - *address DIR*: dirección IP tal como se especificó con el comando "ifconfig" o con "ip a a".
  - netmask MÁSC: máscara de red tal como se especificó con el comando "ifconfig" o con "ip a a".
  - broadcast BRD: dirección broadcast tal como se especificó con el comando "ifconfig".
  - gateway GW: puerta de enlace tal como se especificó con el comando "route add default gw" o con "ip ro a via".
- Opciones para <flia_dirección_inet> = "dhcp"
  - Sin opciones debería funcionar bien.

- Comandos *ifup* e *ifdown*: levantan y bajan (respectivamente) una interfaz. Opciones:
  - { -a | INTERFAZ }: la primera opción levanta/baja todas las interfaces marcadas con "auto" en el archivo /etc/network/interfaces, y la segunda sólo la interfaz especificada.

Luego de hacer cambios en el archivo /etc/network/interfaces se recomienda bajar y luego levantar las interfaces modificadas para que los cambios surtan efecto.

Configuración permanente (sobre archivos) para distribuciones basadas en Gentoo

Se edita el archivo */etc/conf.d/net*. Las líneas que comienzan con "#" son ignoradas.

- Configuración de IP estática: suponiendo que se configurará la interfaz etho con la IP

172.16.0.4, la máscara de red 255.255.0.0, la dirección broadcast 172.16.255.255 y la pasarela 172.16.0.3, las dos siguientes líneas deberían estar escritas en el archivo:

```
config_eth0="172.16.0.4 netmask 255.255.0.0 brd 172.16.255.255"
routes_eth0="default gw 172.16.0.3"
```

- Configuración DHCP: si la interfaz eth0 obtiene sus parámetros de configuración de red por DHCP, entonces debería estar escrita la siguiente línea:

```
config_eth0=”dhcp”
```

Se sugiere la lectura de */etc/conf.d/net.example* para una mejor y más amplia comprensión de este archivo.

9.2.3 . Bajar una interfaz de red

Esto puede hacerse con el comando ifconfig de la siguiente forma

```
diego0 ~ # ifconfig eth0 down
diego0 ~ # _
```

O también con ip

```
diego0 ~ # ip a del 172.16.0.30/16 dev eth0
diego0 ~ # _
```

9.2.4 . Archivo /etc/hosts

Este archivo describe un mapeado de alias de direcciones para los hosts locales que comparten este archivo. Se encuentra en las distribuciones más comunes.

Las líneas que comienzan con el símbolo “numeral” se ignoran. Cualquier otra línea es una lista separada por blancos que comienzan con una IP y luego los dominios a resolver.

9.2.5 . Archivo /etc/host.conf

Consiste de dos líneas (tradicionalmente). La primera es la línea “order”, que indica el orden de búsqueda para la resolución de dominios. Valores válidos para este campo son “hosts”, “bind” y “nis”. El primer valor indica que primero se buscará en el archivo /etc/hosts, y el segundo que se usará DNS. Una línea “order” válida puede ser

```
order hosts, bind
```

Lo que quiere decir que para resolver un dominio primero buscará los alias en el archivo /etc/hosts y, si, no lo encontrara, usará el ServicioDNS para resolverlo.

La segunda línea es la “multi”, sus posibles valores son uno de “on” u “off”, y establece si un mismo dominio puede resolverse como más de una IP (en caso de “on”) o si ser resuelve como la primera IP que encuentra (en caso de “off”) si es que en el archivo /etc/hosts se repitiera para

diferentes IP el mismo valor de dominio.

9.3 . Puertos de red

Un puerto de red es una construcción de software particular para una aplicación o proceso que se comunica en una red. Se identifica con su número (número de puerto), la IP asociada a él y el protocolo usado para la comunicación (comúnmente TCP o UDP).

El número es un entero que va desde cero a 65535 (2^{16} combinaciones). Un proceso es asociado con un puerto en particular (esto es conocido como "binding" o "ligadura", "enlace") para el intercambio de datos, lo que significa que el proceso *estará a la escucha* o *estará escuchando* ("listen" o "listening") a la espera de paquetes entrantes cuyos puertos y direcciones IP destinatarios sean los de *ese* puerto, y/o emitir paquetes salientes cuyos puertos de origen sean *ese* puerto. Los procesos pueden estar ligados a varios puertos.

Las aplicaciones que implementan servicios comunes generalmente están ligadas a puertos específicos definidos por convención. Generalmente serán puertos bajos.

En Linux el comando *netstat* puede mostrar una instantánea de los puertos que están escuchando y los que tienen una conexión establecida. Las siguientes opciones son válidas:

- --all, -a: muestra todos los sockets: los que están escuchando y los que no.
- --numeric, -n: no resolver dominios.
- --program, -p: muestra el PID del proceso al que pertenece el socket.
- -t: mostrar sólo los sockets con protocolo TCP.
- -u: mostrar sólo los sockets con protocolo UDP.
- --verbose, -v: modo verboso.

Ejemplo

```
diego0 ~ # netstat -tuanpv | grep -nE "*"
1:Active Internet connections (servers and established)
2:Proto Recv-Q Send-Q Local Address           Foreign Address
State       PID/Program name
3:tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      4911/sshd
4:tcp        0      0 127.0.0.1:631           0.0.0.0:*
LISTEN      4907/cupsd
5:tcp        0      0 172.16.0.4:44225        64.4.36.31:1863
ESTABLISHED 5137/pidgin
6:tcp        0      0 172.16.0.4:50764        207.46.124.171:1863
ESTABLISHED 5137/pidgin
7:udp        0      0 0.0.0.0:631             0.0.0.0:*
4907/cupsd
diego0 ~ # _
```

9.3.1 . Archivo /etc/services

Este archivo contiene un mapeado entre los nombres "amigables" en forma de texto para los

servicios de internet y sus respectivos números de puerto y protocolo. Por defecto, contiene el mapeado estándar asignado por la IANA (“Internet Assigned Numbers Authority” o “Autoridad de Números de Internet Asignados”).

Cada línea describe un servicio, y tiene la forma

```
<nombre_servicio> <puerto>{ / | , }<protocolo> [alias ...]
```

donde

- <nombre_servicio>: es el nombre amigable del servicio bajo el cual es conocido y buscado. Sensible a la capitalización. A menudo el programa cliente es nombrado después de esto.
- <puerto>: es el número de puerto (en base diez) que el usa servicio.
- <protocolo>: tipo de protocolo usado. Debe ser un de los descritos en */etc/protocols*. Valores típicos son “tcp” y “udp”.
- [alias ...]: es una lista opcional separada por blancos de nombres alternativos para el servicio. También sensible a la capitalización.

Ejemplo

```
diego0 ~ # head -n 56 /etc/services | tail -n 10 | grep -nE "*"
1:chargen               19/tcp          ttytst source   # Character
Generator
2:chargen               19/udp          ttytst source
3:ftp-data      20/tcp                          # File Transfer
[Default Data]
4:ftp-data      20/udp
5:ftp           21/tcp                          # File Transfer
[Control]
6:ftp           21/udp          fsp fspd
7:ssh           22/tcp                          # SSH Remote Login
Protocol
8:ssh           22/udp
9:telnet                23/tcp                          # Telnet
10:telnet               23/udp
diego0 ~ # _
```

9.4 . OpenSSH

OpenSSH (“Open Secure Shell” o “Intérprete Seguro Abierto”) es una suite de aplicaciones que brindan un soporte cifrado para la comunicación en una red, usando el protocolo SSH.

Algunas aplicaciones de la suite son: ssh (cliente de shell de acceso remoto), scp (cliente de copia de archivos remotos), sftp (cliente ftp), sshd (demonio servidor de SSH).

9.4.1 . sshd

Este demonio tiene que estar corriendo para que otros equipos puedan conectarse a él por

medio de los programas *cliente* de la suite. Para correrlo puede ejecutarse *sshd* en una terminal, y también se puede agregar el script de arranque (que viene preparado para la mayoría, si no todas, las distribuciones) al nivel de arranque que se requiera para que se ejecute automáticamente.

9.4.2 . ssh

Sirve para loguearse en un equipo remoto y ejecutar comandos. El comando tiene la siguiente sintaxis:

```
ssh [OTRAS_OPCIONES] [-p puerto_remoto] [-l usuario_remoto]
[usuario_remoto@]NOMBREHOST [comando]
```

Descripción:

- *NOMBREHOST*: nombre del host al que se está por conectar.
- -p PUERTO_REMOTO: número del puerto en el que está escuchando el demonio ssh en el host remoto y por el cual se realizará la conexión. Por defecto (y convención) se usará el puerto 22.
- *-l USUARIO, USUARIO@*: nombre del usuario en el equipo remoto con el cual se está por loguear. Si se usa la segunda forma debe estar inmediatamente antes del nombre del host. Si no se especifica un usuario entonces intentará loguear con el mismo usuario que está ejecutando *ssh* (si en el host no hay un usuario con ese nombre, obviamente no se podrá llevar a cabo el logueo).
- *[comando]*: si se especifica, es el comando que se ejecutará en el host remoto en vez un shell de login.

9.4.3 . scp

Copia archivos entre hosts de una red. Su sintaxis es:

```
scp [OTRAS_OPCIONES] [-pqrv] [-l límite] [-P puerto]
[[usuario_fuente1@]host_fuente1:]archivo1 [...
[[usuario_fuenteN@]host_fuenteN:]archivoN]
[[usuario_destino@]host_destino:]destino
```

Descripción:

La interpretación de “usuario@host” es la misma que para ssh. Si se especifica, debe estar seguido en forma inmediata de “:” y el camino absoluto del archivo (o directorio) que se está especificando.

- -p: preservar los tiempos de acceso y modificación y los modos de los archivos originales.
- -q: modo silencioso (se suprimen las barras de progreso para cada archivo).
- -r: recorrer recursivamente directorios.
- -v: modo verboso.
- -l LÍMITE: límite, en Kb/s, del ancho de banda que se va a usar.
- -P PUERTO: puerto en el que escucha el demonio ssh en el host remoto.

9.4.4 . Revisión general

La primera vez que se ejecuta un programa de la suite se crea una clave para el host local. Con esta clave se identificará este host cuando quiera acceder a otro host. Antes de conectarse con un host nuevo, ssh da aviso que el host no es conocido y pregunta si realmente se desea continuar con la conexión. Si es así, se agrega la clave del host remoto como una línea más en el archivo **~/.ssh/known_hosts**, conteniendo la IP y la clave del host remoto. Este archivo sirve para que la conexión sea más rápida en las próximas ocasiones, no teniendo luego que confirmar la voluntad de conectarse con un mismo host. Como se puede ver, el archivo se encuentra en el directorio personal de cada usuario, por lo que cada usuario puede "confiar" o no en un mismo host en forma independiente. Una vez comenzada la conexión, hay que introducir la clave del usuario con el que se está logueando en el host remoto para continuar.

Puede configurarse ssh para que recuerde los parámetros de los hosts más comunes, editando el archivo **~/.ssh/config**.

Por ejemplo: la conexión con los hosts *jose* y *adrian* es muy frecuente. Se sabe que el demonio ssh en *adrian* no escucha en el puerto por convención (el puerto 22), sino en el 9999. Se sabe también que las respectivas IP de *adrian* y *jose* son 172.16.0.9 y 172.16.0.10. Otro dato que queremos agregar es el usuario con el que vamos a loguearnos en esos hosts, que serán, para *adrian*, el usuario "diego", y, para *jose*, el usuario "jose".

```
diego0 ~ # vi .ssh/config
     .
     .
     .
host adrian
     Hostname 172.16.0.9
     Port 9999
     User diego
host adrian
     Hostname 172.16.0.10
     User jose
     .
     .    SALIR GUARDANDO LOS CAMBIOS
     .
diego0 ~ # _
```

9.5. xinetd

xinetd ("eXtended InterNET services Daemon" o "Demonio de Servicios de Internet Extendido") es un programa usado para mejorar el rendimiento de ciertos programas que proveen servicios de red al mantener esos programas sin ejecutar, escuchando en los puertos en los que esos programas escuchan y ejecutándolos sólo cuando aparece una conexión entrante por ese puerto. Para ejemplificar, usaremos el demonio ssh.

Existen archivos en /etc/ que configuran el comportamiento de xinetd: ellos son **/etc/xinetd.conf** y los archivos en el directorio **/etc/xinetd.d/**, y, respectivamente, configuran la forma de actuar predeterminada de xinetd en relación a su comportamiento genérico y a los servicios que levanta oportunamente.

Cada archivo en el directorio /etc/xinet.d/ contiene la configuración necesaria para levantar un servicio. La estructura genérica de cualquiera de estos archivos es más o menos

```
### <<<<¡Comentario!>>>>
service <nom_servicio>
{
# Habilitación/deshabilitación rápida del servicio
disable          = <{yes/no}>

# Tipo de socket, que debería ser uno de “stream”, “dgram”, “raw”,
# “rdm” o “seqpacket”.
socket_type      = <tipo_de_socket>

# Nombre del servidor (programa a ejecutar), y sus respectivos
# argumentos.
server           = <nom_servidor>
server_args      = <args_servidor>

# Nombre del usuario que ejecutará el servicio
user             = <usuario_local>

# Si es seteado a "yes" el servicio es monohebra y sólo puede
# administrar una conexión de este tipo. Si es seteado a "no" entonces
# por cada servicio nuevo solicitado se arranca un nuevo servidor
# hasta llegar al máximo definido.
wait             = <{yes/no}>

# Cantidad de instancias del servicio permitidas
instances        = <cantidad>

# Puerto (número de puerto) en el que se escuchará en busca de
# conexiones entrantes para el servicio
port             = <núm_puerto>

# Protocolo. Debe existir en /etc/protocols y si no se especifica se
# usa el protocolo por defecto.

# Hosts remotos permitidos. Puede ser, por ejemplo, una lista separada
# por espacios de direcciones IP de hosts y/o redes. Ejemplo:
#  only_from     = 172.16.0.0/16 # Sólo hosts de esta red
#  only_from     = 0.0.0.0 # Cualquier host
only_from        = <hosts>

# Clientes que no tienen acceso. Mismo formato que la opción anterior.
no_access        = <hosts>

# Horarios en los que se puede acceder al servicio. Se usa el formato
# de 24hs y puede ser una lista separada por espacios. Ejemplo:
#  access_times  = 9:00-13:30 16:30-21:00
```

```
access_times      = <horarios_de_acceso>
}
```

A continuación se creará el archivo /etc/xinet.d/sshd para que xinetd lo administre.

```
diego0 ~ # vi /etc/xinetd.d/sshd
     .
     .
     .
service sshd
{
disable     = no
socket_type = stream
wait        = no
user        = root
group       = root
server      = /usr/sbin/sshd
server_args = -if /etc/ssh/sshd_config
protocol    = tcp
port        = 22
type        = UNLISTED
}
     .
     .    SALIR GUARDANDO LOS CAMBIOS
     .
diego0 ~ # _
```

Antes de dejar que xinetd administre las conexiones de sshd, hay que bajar sshd para que no haya problemas, pero primero es interesante ver lo siguiente:

```
diego0 ~ # netstat -tuanpv | grep :22
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      22607/sshd
diego0 ~ # ps xa | grep ssh | grep -v grep
22884 ?        Ss     0:00 /usr/sbin/sshd
diego0 ~ # /etc/init.d/sshd stop
     .
     .
     .
diego0 ~ # _
```

Siempre que se quiera que surtan efecto cambios que recién se hicieron (como la creación de este archivo) se debe reiniciar el demonio xinet. Nótese la nueva salida de *netstat* y de *ps*.

```
diego0 ~ # /etc/init.d/xinetd restart
     .
     .
     .
```

```
diego0 ~ # netstat -tuanpv | grep :22
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      22766/xinetd
diego0 ~ # ps xa | grep ssh | grep -v grep
diego0 ~ # _
```

Ahora es conveniente probar si realmene funciona el trabajo hecho, para ello se puede intentar conectarse al host local vía ssh. Nótese nuevamente las respectivas salidas de los comandos *netstat*, *ps* y también de *tty*.

```
diego0 ~ # ssh localhost
Password:
Last login: Tue May  5 11:07:41 2009 from localhost
diego0 ~ # netstat -tuanpv | grep -n :22
4:tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN      22977/xinetd
6:tcp        0      0 127.0.0.1:57371         127.0.0.1:22
ESTABLISHED 22990/ssh
17:tcp        0      0 127.0.0.1:22            127.0.0.1:57371
ESTABLISHED 22991/5
diego0 ~ # ps xa | grep ssh | grep -v grep
22990 pts/0    S+     0:00 ssh localhost
22991 ?        Ss     0:00 sshd: root@pts/5
diego0 ~ # tty
/dev/pts/5
diego0 ~ # _
```

Con estas salidas se pretende que el lector saque sus propias conclusiones.

9.6 . Descarga en modo texto con los comandos wget y axel

Estos comandos son administradores de descarga. La versión GNU del primero es una herramienta potentísima y dispone de muchas opciones. La segunda es un poco más liviana.

wget

Administrador de descargas no interactivo. Soporta los protocolos HTTP, HTTPS y FTP. Opciones:

- -c, --continue: continua descargando un archivo parcialmente descargado con anterioridad por otra instancia de wget u otro programa.
- --limit-rate=CANT: limita la velocidad de descarga a CANT B/s (por defecto), con el sufijo “k” se mide en KB/s y con “m” en MB/s. Se pueden ingresar valores decimales como “1.2k”.
- --progress=TIPO: tipo de indicador de progreso. Valores posibles para TIPO son “dot” (punto) y “bar” (barra).
- -P DIR, --directory-prefix=DIR: directorio donde se guardan las descargas. Por defecto es el directorio actual.

- [--passive-ftp | --no-passive-ftp]: activa/desactiva el uso del modo de transferencia ftp pasivo.
- -q, --quiet: apaga la salida de wget.
- -t NUM, --tries=NUM: cantidad de reintentos. Cero o “inf” para reintentar siempre. Por defecto hace hasta 20 reintentos.

axel

Descarga archivos de servidores HTTP o FTP a través de múltiples conexiones, y cada una descarga una parte del archivo. El argumento que no se puede evitar es la URL del archivo para descargar. Opciones:

- --max-speed=VELOCIDAD, -s VELOCIDAD: velocidad promedio en B/s a la que debe descargar axel.
- --num-connections=NUM, -n NUM: cantidad de conexiones.
- --output=SALIDA, -o SALIDA: los datos se guardarán en el archivo con nombre SALIDA o en el directorio SALIDA con el mismo nombre que el archivo remoto. Por defecto se guarda en el directorio actual con el mismo nombre que el archivo remoto.
- --verbose: muestra más mensajes de estado. Se puede usar más de una vez para incrementar la verbosidad.
- --quiet, -q: no imprime nada a la Salida Estándar.
- --alternate, -a: muestra un indicador de progreso alternativo.

10 . CUPS: administración de impresoras y trabajos de impresión

10.1 . Introducción

CUPS ("Common UNIX Printing System" o "Sistema Común de Impresión de UNIX") es un sistema modular de impresión que permite a Linux actuar como un servidor de impresión.

Una vez corriendo el demonio cups, puede administrarse fácil e intuitivamente impresoras y trabajos de impresión vía la interfaz web que provee CUPS en el puerto 631 (en general) del host local. Es decir, esta interfaz web de administración puede accederse desde cualquier navegador web ingresando la dirección *127.0.0.1:631* ó *localhost:631*.

En general, el demonio cups puede necesitar ajustes; en esos casos es probable que se tenga que hacer algún cambio en el archivo de configuración **/etc/cups/cupsd.conf**.

10.2 . Manipulación de trabajos de impresión

Comando lp

Imprime archivos y altera trabajos pendientes. Se puede usar "-" como nombre de archivo para forzar la impresión desde la Entrada Estándar. Opciones:

- -n CANT: cantidad de copias que se imprimirán.
- -q PRIOR: especifica la prioridad del trabajo. Ésta va desde 1 (la más baja) hasta 100 (la más alta). Por defecto será de 50.
- -t "NOMBRE": establece el nombre del trabajo.
- -u NOM_USUARIO: envía un trabajo como el usuario NOM_USUARIO.
- -H VALOR: determina cuándo debe imprimirse un trabajo. Valores posibles son:
  - HH:MM: determina la hora exacta en que se imprimirá el trabajo.
  - hold: pausa indefinidamente un trabajo.
  - imediate: imprimirá el documento de inmediato.
  - restart: junto con la opción -i reinicia un trabajo terminado.
  - resume: junto con la opción -i reanuda un trabajo pausado.
- -P LISTA_DE_PÁGINAS: especifica qué páginas de un documento imprimir. Valores válidos puede ser una lista separada por comas de números, que indica que se imprimirán esas páginas. También se puede especificar dos números separados por guiones regulares que indican que se imprimirán las páginas en ese intervalo; y también puede ser una combinación de los dos anteriores.

Comando lpinfo

Lista los dispositivos o controladores conocidos por el servidor CUPS. Opciones:

- -v: lista dispositivos.
- -m: lista controladores.

Comando lpq

Muestra el estado de la cola de impresión.

Comando cancel

Cancel trabajos de impresión existentes. Con la opción *-a* cancela todos los trabajos. Recibe como argumento el id del trabajo de impresión.

Comando lpstat

Muestra información sobre las impresoras y trabajos de impresión actuales. Si se corre sin argumentos lista los trabajos de impresión encolados.

11 . Tareas y configuraciones varias

11.1 . Configuración de video

En Linux, existen por lo menos tres capas de software que brindan soporte para el entorno visual: el sistema de ventanas (X.Org), el administrador de ventanas y el entorno de escritorio (estos dos generalmente englobados por KDE y GNOME). Es importante saber distinguir en Linux el SO en sí y la parte visual. El entorno visual es solamente una aplicación que corre sobre el SO y es totalmente independiente de éste. Si se cuelga la parte visual se puede igualmente trabajar en otras terminales y realizar tareas. De hecho, *el sistema visual corre sobre una terminal*, generalmente en la tty7, y se puede tener más de un entorno visual corriendo en terminales diferentes.

X.Org es una implementación de código abierto del Sistema de Ventana X ("X Window System") y sirve para brindar una interfaz gráfica a sistemas como Linux. El servidor provee acceso a la pantalla, teclado y ratón, mientras que los clientes, las aplicaciones, usan estos recursos para interactuar con el usuario. De este modo, mientras el servidor se ejecuta de manera local, puede haber aplicaciones remotas en otras máquinas que usen este servidor.

KDE ("K Desktop Environment") y GNOME ("GNU Network Object Model Environment") son entornos de escritorio e infraestructuras de desarrollo para sistemas como Linux. Es importante saber que se puede tener instalados ambos en un mismo equipo y se los puede ejecutar en forma independiente sin problemas de compatibilidad. También se pueden ejecutar aplicaciones KDE en GNOME (y viceversa) si es que se tienen las librerías indicadas.

Reconfiguración de video

Hoy en día es poco común que haya que hacer configuraciones visuales en sistemas Linux modernos ya que poseen una buena detección de hardware y una buena configuración automatizada.

Sin embargo, es posible que se quiera portar el disco rígido a otro equipo y arrancar desde él, entonces habría que contar con ciertas precauciones; por ejemplo, nuevamente que estén compilados dentro del kernel los controladores necesarios, etc. Otra consideración es que es probable que el entorno gráfico se cuelgue pues la configuración del servidor X está orientada a otro hardware, por lo que se puede correr el programa que se muestra a continuación. Pero antes, hay que asegurarse que el servidor X no está corriendo. Esto se puede verificar viendo la terminal 7. Si no está corriendo, tendría que haber un fondo como el de una terminal común. También se puede usar el comando *ps* de la siguiente forma

```
diego0 ~ # ps xa | grep X | grep -v grep
 4986 tty7     Ss+    6:10 /usr/bin/X -br -nolisten tcp :0 vt7 -auth /
var/run/xauth/A:0-IzfIPZ
diego0 ~ # _
```

Esto quiere decir que el servidor está corriendo.

Para terminar su ejecución se puede utilizar la combinación de teclas CTRL+ALT+RETROCESO. Esto terminará la ejecución del servidor X y de cualquier entorno de

escritorio (como KDE o GNOME) que haga uso de él, pero generalmente el *servicio de escritorio*, ubicado en /etc/init.d/ bajo el nombre *gdm* para GNOME y *kdm* para KDE, se levantará de nuevo junto con el servidor X. Esto se puede solucionar deteniendo el servicio correspondiente con el parámetro *stop*.

Y ahora la configuración automática

```
diego0 ~ # Xorg -configure
     .
     .
     .
Module Loader present
Markers: (--) probed, (**) from config file, (==) default setting,
        (++) from command line, (!!) notice, (II) informational,
        (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Thu May  7 11:53:54 2009
List of video drivers:
        tga
        i128
        trident
     .
     .
     .
        fbdev
        vesa
        vga
(++) Using config file: "/root/xorg.conf.new"
(II) Module "ddc" already built-in


Xorg detected your mouse at device /dev/input/mice.
Please check your config if the mouse is still not
operational, as by default Xorg tries to autodetect
the protocol.

Your xorg.conf file is /root/xorg.conf.new

To test the server, run 'X -config /root/xorg.conf.new'

diego0 ~ # _
```

Con este comando el servidor X se autoconfigura, probando los módulos controladores de video, y escribe un nuevo archivo de configuración en el directorio personal del usuario que lo ejecutó con el nombre *~/xorg.conf.new*. Finalmente se puede probar que realmente funciona esta configuración ejecutando el comando propuesto

```
diego0 ~ # X -config /root/xorg.conf.new
     .
     .
```

.

Si todo anda bien, se tendría que ver algo así como un fondo gris plomo punteado y un puntero de mouse con forma de equis. Se trata del servidor X corriendo solo, sin ningún entorno de escritorio. Si esta configuración es buena, entonces se puede renombrar el archivo de configuración por defecto /etc/X11/xorg.conf y copiar el nuevo a ese nombre. Antes, terminamos el servidor X con la combinación de teclas.

.
.
.

```
diego0 ~ # mv /etc/X11/xorg.conf /etc/X11/xorg.conf.maquinavieja
diego0 ~ # cp /root/xorg.conf /etc/X11/xorg.conf
diego0 ~ # _
```

Y así se dispone de dos configuraciones para el servidor X, una para cada máquina.

11.2 . Tareas programadas

Para realizar tareas de este tipo es posible hacer uso de la herramienta *cron* (“Cronograph” o “Cronógrafo”). Este software puede encontrarse en general como *vixie-cron* y *cronbase*.

Cómo correr tareas programadas

Una vez instalado el demonio cron, cuyo initscript puede llamarse /etc/init.d/vixie-cron, hay que asegurarse que esté corriendo para que ejecute las tareas que se le especifican, y luego editar, como superusuario, la configuración de las tareas.

Crear tareas

Las tareas deben crearse con el comando *crontab -e*, que edita un archivo temporal (preferentemente von *vi*) en el cual se pueden especificar variables simples y tareas. Las variables se especifican en la misma forma que en bash, y las líneas que contienen tareas están compuestas por seis campos separados por blancos: “Minutos” (de cero a 59), “Horas” (de cero a 23), “Días del mes” (de uno a 31), “Meses” (de uno a 12), “Días de la semana” (de cero a 7, cero y 7 son Domingo) y “Tarea”. Los primeros cinco se refieren al momento en que se ejecutará “Tarea”, y queda claro que la menor porción de tiempo determinable es el minuto.

En los primeros cinco campos puede especificarse, además de sólo un número, lo siguiente

- Rangos de dos números, separados por un guión regular, entendiéndose como *todos los valores entre esos extremos y los extremos mismos*.
- Listas de valores o de rangos, cada elemento separado por una coma.
- Sólo un asterisco en algún campo, lo que equivale a una lista de todos sus posibles valores. Por ejemplo, un asterisco en “Meses” es lo mismo que escribir “1-12”.
- Valores de *paso*, de regularidad o de periodicidad, especificados luego de un rango. Se escribe luego del rango una barra de división y luego el valor de periodicidad, significando que sólo se toman los valores en el rango que están separado de N cantidad de períodos del

extremo inferior. Por ejemplo, si para el campo “Minutos” se escribe “10-31/3” es lo mismo que “10,13,16,19,22,25,28,31”, o si se escribe “*/9” es lo mismo que “cada 9 minutos”.

En lugar de los primeros cinco campos puede escribirse cualquiera de las siguientes cadenas especiales.

- “@reboot”: correr durante el buteo.
- “@yearly”, ”@annually”: una vez al año; equivalente de “0 0 1 1 *”.
- “@monthly”: una vez al mes; equivalente de “0 0 1 * *”.
- “@weekly”: una vez a la semana; equivalente de “0 0 * * 0”.
- “@daily”, “@midnight”: una vez al día; equivalente de "0 0 * * *".
- “@hourly”: una vez cada hora; equivalente de "0 * * * *".

Revisión general

Cada usuario puede tener su propio crontab, es decir, sus propias tareas programadas, pero las tareas del sistema debe programarlas el administrador logueado como el superusuario.

Puede listarse todas las tareas programadas para un usuario con el comando *crontab -l*.

```
diego0 ~ # crontab -l
# (/tmp/crontab.XXXXJmcBWX installed on Wed Dec  5 12:33:28 2007)
# (Cron version V5.0 -- $Id: crontab.c,v 1.12 2004/01/23 18:56:42
vixie Exp $)
*/5 * * * * /usr/bin/fetchmail
0 */1 * * * /root/bin/autom
*/5 * * * * /bin/rutear
diego0 ~ # _
```