



Gobierno **Bolivariano**  
de Venezuela

Ministerio del Poder Popular  
para las **Telecomunicaciones y la Informática**



# SOPORTE TÉCNICO GNU/LINUX





Av. Universidad, Esq. El Chorro, Torre Ministerial, Piso 12, La Hoyada, ZP 1010, Caracas - Venezuela  
Master: +58 (212) 771.8800 / Rif: G-2000-4417-9 / Sitio web: [www.cnti.gob.ve](http://www.cnti.gob.ve)






## **Reconocimiento-No comercial-Compartir bajo la misma licencia 3.0**

Usted es libre de:

-  copiar, distribuir y reproducir públicamente la obra
-  hacer obras derivadas

Bajo las siguientes condiciones:

-  **Reconocimiento.** Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).
-  **No comercial.** No puede utilizar esta obra para fines comerciales.
-  **Compartir bajo la misma licencia.** Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.
  - Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
  - Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
  - Nada en esta licencia menoscaba o restringe los derechos morales del autor.

**Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.**

Esto es un resumen fácilmente legible del texto legal de versión original en Idioma Inglés (la licencia completa)

<http://creativecommons.org/licenses/by-nc-sa/3.0/ec/legalcode>

## ÍNDICE DE CONTENIDOS

INTRODUCCIÓN.....	6
UNIDAD I - INSTALACIÓN DE GNU/LINUX - DISTRIBUCIÓN CANAIMA.....	9
PREVIOS A LA INSTALACIÓN DE GNU/LINUX.....	9
PROCESO DE INSTALACIÓN DE GNU/LINUX - DISTRIBUCIÓN CANAIMA.....	14
UNIDAD II - SISTEMA X.Org.....	20
EL SISTEMA X.Org.....	20
SESIONES.....	25
UNIDAD III - EL SHELL.....	27
USANDO EL SHELL.....	27
SHELL SCRIPTS.....	29
USANDO HISTORY.....	30
PROCESOS.....	32
COMANDOS.....	36
UNIDAD IV - GESTIÓN DE USUARIOS Y GRUPOS.....	46
GESTIÓN DE USUARIOS.....	46
GESTIÓN DE GRUPOS .....	50
PERMISOS DE ARCHIVOS .....	51
PERMISOS: SUID y SGID .....	51
UNIDAD V - EDITOR VIM .....	55
INTRODUCCIÓN A VIM.....	55
UNIDAD VI - INSTALACIÓN DE PAQUETES.....	66
ADVANCED PACKAGING TOOL (APT).....	66
APTITUDE	
Aptitude es un envoltorio que trabaja sobre apt. No es gráfico, sino que tiene una interfaz con debconf y también puede usarse en línea de comandos. ¿Qué lo diferencia de apt? .....	68
SYNAPTIC.....	69
DPKG.....	69
DSELECT.....	70
ALIEN.....	70

UNIDAD VII - KERNEL.....	71
DEFINICIÓN DE KERNEL .....	71
UNIDAD VIII - MANEJO Y TIPOS DEL SISTEMA DE ARCHIVOS.....	75
CONSIDERACIONES AL MOMENTO DE HACER UN FILE SYSTEM.....	75
DISPOSITIVOS EN LINUX.....	75
PARTICIONES.....	76
SISTEMA DE ARCHIVOS.....	80
CUOTAS DE DISCO.....	86
UNIDAD IX - FUNDAMENTOS DE REDES TCP/IP.....	90
PROTOCOLO TCP/IP.....	90
PROTOCOLOS DE APLICACIONES.....	90
DIRECCIONES PRIVADAS.....	91
CONFIGURACIÓN DE LA RED.....	92
SECURE SHELL – SSH.....	98
COMANDOS SSH Y SCP.....	100
SERVICIO VNC .....	101
UNIDAD X - SERVICIO CUPS.....	103
FUNCIONAMIENTO DE CUPS.....	103
PPDS.....	104
LA INTERFAZ WEB DE ADMINISTRACIÓN DE CUPS.....	105
IMPRESORAS LOCALES (USB Y PARALELO).....	106
UNIDAD XI: SERVICIO NFS.....	109
DEFINICIÓN DE NFS.....	109
CARACTERÍSTICAS ÚTILES DE NFS.....	109
EL SERVIDOR NFS.....	109
EL CLIENTE NFS.....	113
PRECAUCIONES.....	114
USANDO NIS, NFS Y AUTOFS.....	115
UNIDAD XII - SERVICIO SAMBA.....	117
INTRODUCCIÓN .....	117
INSTALACIÓN DEL SERVIDOR SAMBA .....	117

CONFIGURACIÓN DEL SERVIDOR .....	117
COMANDOS SOBRE EL SERVIDOR .....	118
DANDO DE ALTA USUARIOS .....	119
EL CLIENTE SAMBA .....	120
EJERCICIOS.....	124
EJERCICIOS N° 1.....	124
EJERCICIOS N° 2.....	124
EJERCICIOS N° 3.....	124
EJERCICIOS N° 4.....	125
EJERCICIOS N° 5.....	125
EJERCICIOS N° 6.....	126
EJERCICIOS N° 7.....	127
EJERCICIOS N° 8.....	127
EJERCICIOS N° 9.....	128
EJERCICIOS N° 10.....	129
EJERCICIOS N° 11.....	129
EJERCICIOS N° 12 - Compartiendo un directorio (en toda la red local).....	129
EJERCICIOS N° 13 - Compartiendo un directorio con permiso de lectura y escritura.....	130
EJERCICIOS N° 14 - Opciones de inicio en /etc/fstab.....	130
EJERCICIOS N° 15 - Accediendo con permiso de root (muy peligroso).....	130

## INTRODUCCIÓN

### SISTEMA OPERATIVO GNU/LINUX

Un sistema operativo consiste en varios programas fundamentales que necesita el computador para poder comunicar y recibir instrucciones de los usuarios, tales como: leer y escribir datos en el disco duro e impresoras, controlar el uso de la memoria y ejecutar otros programas. GNU/Linux es un Sistema Operativo, es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo.

Como sistema operativo es muy eficiente y tiene un excelente diseño. Es multitarea, multiusuario, multiplataforma y multiprocesador; en las plataformas Intel corre en modo protegido, protege la memoria para que un programa no pueda hacer caer al resto del sistema, carga sólo las partes de un programa que se usan, comparte la memoria entre programas aumentando la velocidad y disminuyendo el uso de memoria, usa un sistema de memoria virtual por páginas, utiliza toda la memoria libre para caché, permite usar bibliotecas enlazadas tanto estática como dinámicamente, se distribuye con código fuente, usa hasta 64 consolas virtuales, tiene un sistema de archivos avanzado pero puede usar los de los otros sistemas y soporta redes tanto en TCP/IP como en otros protocolos.

En GNU/Linux, Linux es el núcleo y el resto del sistema consiste en otros programas, muchos de los cuales fueron escritos por o para el proyecto GNU. Dado que el núcleo de Linux en sí mismo no forma un sistema operativo funcional, se prefiere utilizar el término *GNU/Linux* para referirse a los sistemas que la mayor parte de las personas llaman de manera informal *Linux*.

La mayor parte del desarrollo de GNU/Linux lo realizan voluntarios de forma altruista, lo que significa que nadie es dueño del sistema como sucede en otros casos; esto le permite tener grandes ventajas, tales como: poder elegir entre docenas de distintos intérpretes de línea de comandos y entre distintos entornos de escritorio. Tantas opciones confunden a veces a los usuarios de otros sistemas operativos que no están acostumbrados a poder modificar el intérprete de línea de comando o el entorno de escritorio. Es menos probable que un sistema GNU/Linux se colapse, además tiene mejor capacidad para ejecutar múltiples programas al mismo tiempo y es más seguro que muchos otros sistemas operativos. Debido a estas ventajas, es el sistema operativo que ha experimentado mayor crecimiento en el mercado de los servidores.

## **DEBIAN GNU/LINUX**

Dentro de lo que se conoce como proyecto GNU/Linux existen múltiples distribuciones con un punto común, el núcleo Linux. Una distribución GNU/Linux es un conjunto de aplicaciones reunidas para permitir la instalación sencilla del sistema, incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios.

Debian GNU/Linux o mejor conocido como Debian, es una de las múltiples distribuciones que hoy en día se puede encontrar en la red trabajando con el núcleo Linux y basada en el proyecto GNU. Actualmente, Debian GNU/Linux ofrece 18733 paquetes, programas precompilados distribuidos en un formato que hace más fácil su instalación.

## **CANAIMA**

Es una distribución GNU/Linux venezolana basada en Debian, excepto por un determinado número de paquetes necesarios para la adaptación a las necesidades locales de la Administración Pública Nacional (APN), de hecho, surge como una solución para cubrir las necesidades ofimáticas de los usuarios finales de la APN y para dar cumplimiento al decreto presidencial Nro. 3.390 sobre el uso de Tecnologías Libres en la APN. Canaima es 100% compatible con Debian y sus paquetes pueden ser actualizados usando los repositorios oficiales de esta última. Entre las características más relevantes de Canaima están:

- Totalmente desarrollada en Software Libre.
- No está limitada al uso en la APN, sino que puede ser usado por cualquier persona.
- Se encuentra equipado con herramientas ofimáticas como OpenOffice, (procesador de palabras, hojas de cálculo, presentaciones), diseño gráfico, planificación de proyectos y bases de datos.
- Permite la interacción con Internet, a través de su navegador web, gestor de correo electrónico y aplicaciones para realizar llamadas telefónicas por la red.
- Es estable y segura, basada en la versión estable de Linux Debian, la cual pasa por una serie de procesos y pruebas rigurosas de calidad.

- Realizada en Venezuela por talento nacional.
- Huso horario nacional actualizado.



## **UNIDAD I - INSTALACIÓN DE GNU/LINUX - DISTRIBUCIÓN CANAIMA**

### **PREVIOS A LA INSTALACIÓN DE GNU/LINUX**

A continuación se describen los pasos a seguir durante el proceso de instalación de cualquier distribución GNU/Linux:

- Realizar una copia de seguridad de los datos o documentación existente en el disco duro donde se planea realizar la instalación.
- Reunir información sobre el sistema, así como toda la documentación que se necesite antes de iniciar la instalación.
- Crear un espacio particionable en el disco duro para la instalación del sistema operativo, de ser necesario.
- Localizar y/o descargar el programa instalador, así como los archivos de cualquier controlador especializado que la computadora donde se va a instalar el sistema necesite.
- Instalar los archivos de arranque (la mayoría de los usuarios de CD pueden arrancar desde uno de éstos).
- Arrancar el sistema de instalación.
- Elegir el idioma para la instalación.
- Activar la conexión de red, si está disponible.
- Crear y montar las particiones en las que se instalará el sistema operativo.
- Esperar a la descarga/instalación/configuración automática del sistema base.
- Instalar el gestor de arranque.

#### **Información del Hardware de la Computadora.**

En la mayoría de los casos, el instalador detecta automáticamente el hardware del computador donde se instala el sistema. Sin embargo, es posible que esto no suceda, si es este el caso, se debe estar

preparado. Por lo tanto, se recomienda estar familiarizado con el hardware de la máquina antes de la instalación. En este sentido, se debe obtener la información del hardware de la computadora, para esto se pueden utilizar:

- Los manuales que vienen con cada pieza de hardware.
- Las pantallas de configuración de la BIOS del computador. Estas pueden verse cuando se enciende la máquina y se presiona una combinación de teclas (verificar el manual para saber la combinación, la mayoría de las veces se utiliza la tecla *Supr* ).
- Las cajas y cubiertas de cada pieza de hardware.
- Órdenes del sistema o herramientas de otros sistemas operativos, incluyendo las capturas de pantallas de los gestores de archivos. Esta fuente de información es especialmente útil para obtener información sobre la memoria RAM y el espacio disponible en el disco duro.
- El administrador de sistemas o proveedor de servicio de Internet puede ofrecer información necesaria para configurar la red y el correo electrónico, ésto si se sistema está conectado a alguna red durante todo el día. Por ejemplo, si utiliza una conexión Ethernet o equivalente, pero no si tiene una conexión PPP (Protocolo Punto a Punto).

<b>HARDWARE</b>	<b>INFORMACIÓN QUE SE PUEDE NECESITAR</b>
Discos duros	El número de discos que tiene.
	Orden en el sistema.
	Si es IDE ó SCSI
	Espacio libre disponible.
	Particiones.
	Particiones con otros sistemas operativos instalados.
Monitor	Modelo y fabricante.
	Resoluciones soportadas.
	Rango de refresco horizontal.
	Rango de refresco vertical.
	Profundidad de color soportada (número de colores).
	Tamaño de la pantalla.
Ratón	Tipo: serie, PS/2 ó USB.
	Puerto.
	Fabricante.
	Número de botones.
Red	Modelo y Fabricante.
	Tipo de adaptador.
Impresora	Modelo y fabricante.
	Resoluciones de impresión soportadas.
Tarjeta de vídeo	Modelo y fabricante.
	Memoria RAM de vídeo disponible.
	Resoluciones e intensidad de colores soportadas .

### **Medios de Instalación**

En esta sección se puede determinar los diferentes tipos de medios que se usan para instalar el sistema operativo GNU/Linux.

- *CD-ROM/DVD-ROM*: existe soporte para la instalación basada en CD-ROM para algunas arquitecturas o para propósitos de recuperación del sistema.
- *Dispositivo de memoria USB*: son utilizados para gestionar (instalar y cuando sea necesario recuperar el sistema) servidores y en los casos de sistemas pequeños que no tienen espacio para unidades innecesarias.
- *Red*: se utiliza durante la instalación para recuperar archivos. El que se utilice la red o no,

depende del mecanismo de instalación que se escoja y de las respuestas dadas a algunas preguntas que se realizarán durante la instalación. Este sistema de instalación puede utilizar la mayor parte de las conexiones de red a través tanto de HTTP como FTP. También se puede arrancar el sistema de instalación a través de la red.

### **Tarjetas de Red Inalámbricas**

En GNU/Linux existe un buen soporte de la mayoría de las tarjetas inalámbricas pero con un factor que se debe tener en cuenta, y es que una gran cantidad de adaptadores inalámbricos han de utilizarse con controladores que o bien no son libres o bien no se han aceptado en el núcleo oficial Linux. Estas tarjetas generalmente pueden configurarse para que funcionen en GNU/Linux, pero no están soportadas durante la instalación.

En algunos casos el controlador que se necesita puede no estar disponible como paquete, en esa situación se debe comprobar si existe código fuente disponible en Internet y compilar el controlador por si mismo. Si no hay algún controlador de Linux disponible se puede utilizar como último recurso el paquete *NDISwrapper* que permite el uso de tarjetas inalámbricas en Sistema Operativo GNU/Linux utilizando el kernel de Windows.

### **Requisitos de Memoria y Espacio en Disco Duro**

En ciertas ocasiones nos puede interesar conocer cuáles son los requisitos de *hardware* que necesitan una u otra distro para su instalación. No siempre es fácil encontrar dichos datos, por lo que se ha recopilado la información y con ella se ha creado una tabla comparativa (<http://www.microtecnologias.cl/blog/?p=904#more-904>) que permite decidir cuál distribución funcionará mejor en la computadora donde se desea instalar el sistema.

En nuestro caso nos interesa las distros Debian y Canaima, por lo tanto la información es la siguiente:

*Debian 3.0:*

- Procesador: Intel Pentium 1-4, AMD Duron, Celeron, Athlon, Semprom u Opteron.
- RAM: Mínimo 16 MB para modo texto, 64 MB interfaz gráfica / Recomendado: 128 MB.
- Espacio en Disco Duro: Mínimo 450 MB / Recomendado 4 GB.

*Debian 3.1:*

- Procesador: Intel Pentium 1-4, AMD Duron, Celeron, Athlon, Semprom u Opteron.
- RAM: Mínimo 32 MB para modo texto, 194 MB interfaz gráfica / Recomendado: 256 MB.
- Espacio en Disco Duro: Mínimo 500 MB / Recomendado 3 GB.

#### *Canaima:*

- Procesador: Basado en Intel x86 i386, mínimo Pentium III.
- RAM: Mínimo 64 MB / Recomendado 512 MB.
- Espacio en Disco Duro: Mínimo 5 GB.

### **Esquema de Particiones**

El particionamiento es la creación de divisiones lógicas en un disco duro que permite aplicar el formato lógico de un sistema operativo específico. Cada partición aparece ante el sistema como si fuese un disco independiente.

Un disco duro puede tener un máximo de 4 particiones primarias, porque la información de la tabla de particiones reside (junto con el código de arranque) en el *MASTER BOOT RECORD (MBR)*: el sector 0 del disco. Sin embargo, una de las particiones primarias puede ser designada como *partición extendida* y ser subdividida en un número ilimitado de particiones *lógicas*.

GNU/Linux puede ser instalado en cualquier tipo de partición y suele numerar las particiones primarias de un disco desde 1 a 4 reservando los números 5 y superior para las particiones lógicas.

Es usual que en los sistemas GNU/Linux se creen hasta 3 particiones: la principal representado por el símbolo / la cual contiene todo el software del Sistema Operativo, una segunda para el directorio *home* que contiene las configuraciones de usuario y una tercera llamada *swap* para la memoria virtual temporal que es utilizada en casos de sobrecarga de trabajo, esto para un esquema simple y efectivo. Si el usuario es avanzado puede necesitar particiones separadas para aplicaciones, archivos temporales, entre otros. Por ejemplo: */usr* para el directorio de aplicaciones, */var* para el directorio de logs y otros archivos de tamaño variable, */tmp* para directorio de archivos temporales y */opt* para directorio de software comercial específico.

## Nomenclatura para Discos y Particiones

En el diseño tradicional UNIX, todo es un fichero y los discos se nombran mediante su fichero de dispositivo: IDE, SCSI y USB.

- *IDE* : /dev/hda Disco Maestro en canal IDE 0, /dev/hdb Disco Esclavo en canal IDE 0, /dev/hdc Disco Maestro en canal IDE 1, /dev/hdb Disco Esclavo en canal IDE 1.
- *SCSI y USB*: /dev/sda, /dev/sdbb, entre otros.

Las particiones de un disco se nombran mediante el nombre de dispositivo y el número de partición:

- *Primarias*: /dev/hda1, /dev/hda2, /dev/hda3, /dev/hda4.
- *Lógicas*: /dev/hda5, entre otras.

## Gestor de Arranque

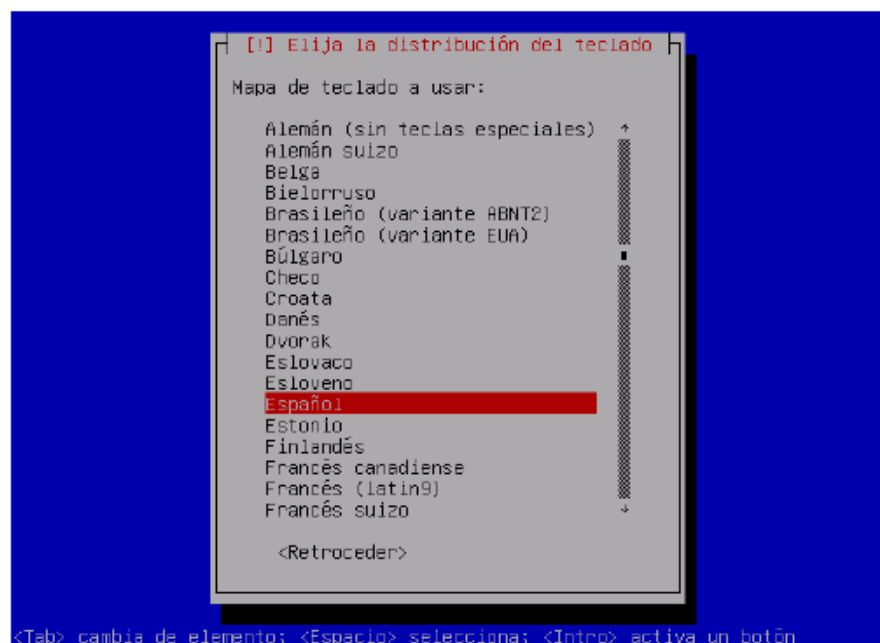
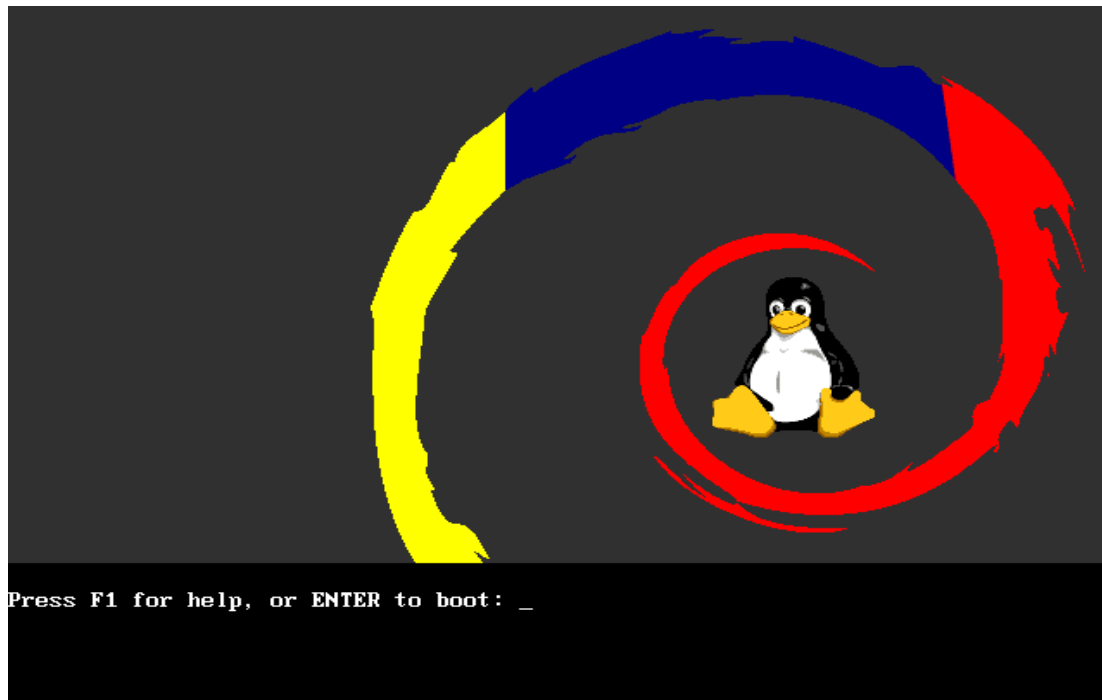
Un gestor de arranque es un programa que se carga en el momento de arrancar el computador y permite elegir qué sistema operativo, de entre los que haya instalados en el disco duro, se quiere iniciar.

Conceptualmente todos los gestores funcionan de la siguiente manera: primero la BIOS del computador debe leer el código de arranque del MBR (sector 0 del disco). Para ello se debe configurar la BIOS para que pueda arrancar del disco que se quiere. La BIOS sólo sabe arrancar el programa que se encuentra en el MBR, dicho programa es el gestor de arranque, en su primera etapa y a su vez sabe a qué particiones tiene que ir a leer para continuar con la carga de la siguiente etapa, y de ahí ofrecer un menú para que el usuario seleccione uno u otro sistema operativo.

Uno de los gestores más flexibles y el que se ha convertido en estándar es *GRUB (GRand Unified Bootloader)* el cual es un gestor de arranque múltiple que se usa comúnmente para iniciar dos o más sistemas operativos instalados en un mismo computador. GRUB viene preinstalado en la mayoría de las distribuciones de GNU/Linux modernas, entre ellas Debian y sus derivadas.

## PROCESO DE INSTALACIÓN DE GNU/LINUX - DISTRIBUCIÓN CANAIMA

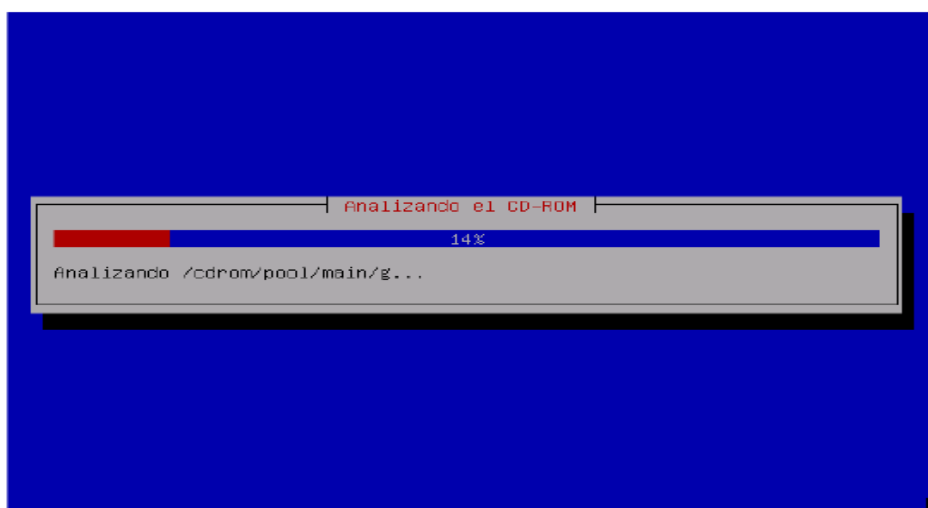
Para instalar Canaima se debe bootear con el CD de instalación, una vez que se inicie el instalador, se mostrará una pantalla inicial que da la bienvenida al sistema, allí se debe pulsar la tecla *Enter*.



Después de unos instantes se solicita elegir el idioma. Por defecto, el idioma seleccionado es “Español”, se puede usar las teclas de desplazamiento para elegir otro idioma si es el caso, luego presionar la tecla *Enter* para continuar.

Las opciones de nuestro país vienen preconfiguradas por defecto, por ejemplo: la zona horaria y la distribución del teclado.

En la siguiente pantalla el instalador verifica el hardware de la computadora y configura la red. Si la máquina está conectada a una red que asigne direcciones IP automáticas, el proceso de instalación lo detectará de manera automática por medio de DHCP (**D**ynamic **H**ost **C**onfiguration **P**rotocol), de lo contrario el usuario debe comunicarse con el administrador de la red quien le proporcionará los datos que se deben colocar para configurar la red en el sistema. Si la máquina no se encuentra conectada a ninguna red, igualmente se solicitará una dirección IP, en tal caso se puede colocar una como por ejemplo: 192.168.0.1 con una máscara 255.255.0.0 sin ninguna pasarela o gateway y se continua con la instalación.



Ahora corresponde particionar los discos. Para ello, el sistema instalador le brinda al usuario la oportunidad de particionar automáticamente o de manera manual el disco. En este sentido se presentan las siguientes opciones:

- *Guiado- Utilizar todo el disco:* recomendado para usuarios noveles, propone un particionado adecuado de forma automática y utilizando todo el disco, al seleccionar esta opción se mostrará otro menú para escoger entre todos los archivos en una partición. Las opciones del menú son:



- ➔ Crear solo 2 particiones, una para el área de intercambio o swap y la otra para el sistema de archivos raíz ( / ) de donde se crean los demás directorios y archivos del sistema.
- ➔ Separar la partición /home, esta opción permite separar la partición /home de la partición raíz ( / ), esto tiene la ventaja de que los directorios y archivos de los usuarios quedan separados y a la hora de que por algún motivo se tenga que rehacer el sistema, los datos de los usuarios quedarán en una partición aparte y no se tendrá la necesidad de formatear la misma, ya que únicamente se trabajará con la partición raíz ( / ).
- Separar particiones /home, /usr, /var y /tmp, ésta opción requiere de un mayor conocimiento sobre el sistema de archivos en Linux y no es recomendado para usuarios noveYuddeliales.
- *Guiado - utilizar el disco completo y configurar LVM (Logical Volume Management)*: al igual que en la opción anterior propone de forma automática y utilizando todo el disco, el particionamiento adecuado y, además, permite configurar LVM. LVM permite agrupar discos físicos en grupos virtuales de discos y posteriormente crear particiones o volúmenes lógicos.
- *Guiado - utilizar el disco completo y configurar LVM cifrado*: funciona de igual manera que la opción anterior, pero adicionalmente cifra los datos.
- *Manual*: particionamiento completamente manual, es recomendable para usuarios avanzados. Con esta opción el usuario debe crear todas las unidades necesarias (/, swap, ext3, entre otras) manualmente. Luego de seleccionar el particionado de disco y continuar, aparece una pantalla de confirmación donde se pueden ver las particiones que se van a crear, si se está de acuerdo se escoge la opción *Sí*, de lo contrario la opción *No*. Si la opción escogida es *No* se selecciona nuevamente el tipo de particionado para elaborar las particiones como se desean.

Es importante resaltar que una vez que se es seleccionada la opción *Sí*, los cambios realizados no se pueden deshacer, por lo tanto se debe estar seguro del particionado que se está realizando.

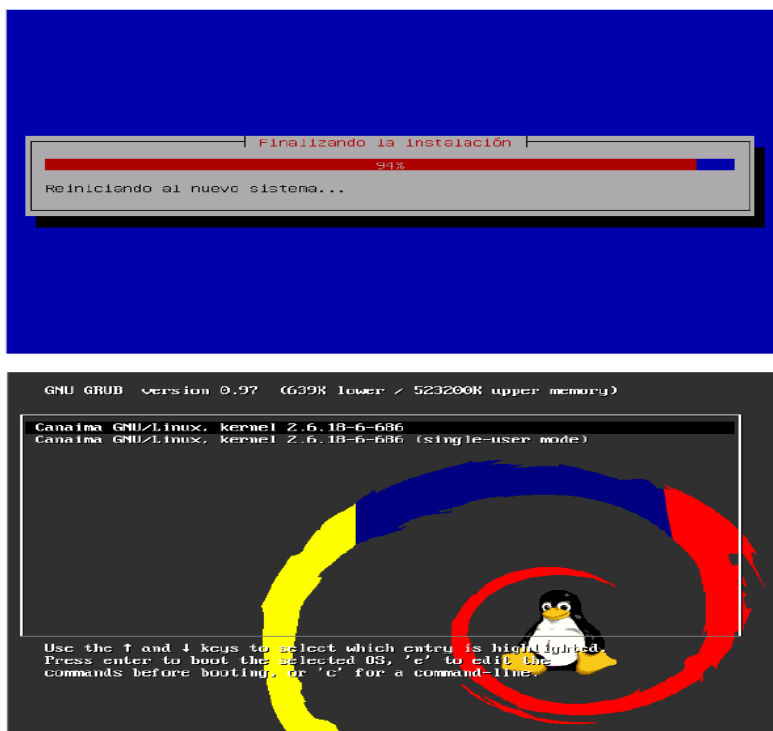
A continuación se solicitará lo siguiente:

- Nombre para el usuario del computador. Por ejemplo: Rafael Méndez.
- Nombre para la cuenta de usuario. Por ejemplo rmendez
- Contraseña, la cual debe ser verificada por el usuario.

A partir de ese momento el programa instalador formatea las particiones y empieza a instalar el sistema base, lo que puede tomar un tiempo. Tras esto se llevará a cabo la instalación del núcleo. El último paso es la instalación del gestor de arranque, el cual es agregado automáticamente por el instalador y éste mostrará un aviso si detecta otros sistemas operativos en el computador. GRUB se instala de forma predeterminada en el sector de arranque del primer disco duro. Sin embargo, se puede cambiar e instalar en otra ubicación si así se desea.

Luego el programa instalador pregunta si se quiere usar una réplica de red, si se selecciona *Sí* se agregan los repositorios de preferencia del usuario, por defecto la distribución Canaima tiene configurados los repositorios del Centro Nacional de Tecnologías de Información (CNTI) por lo que no es necesario ingresarlos en este punto, luego pregunta si se dispone de un servidor proxy, si se selecciona la opción *No* igualmente se instalará el sistema con entorno gráfico, los paquetes básicos y adicionales que se encuentran en el CD.

Ahora el programa instalador indicará que la instalación ha finalizado, se debe retirar el CD de instalación y pulsar la tecla *Enter* para reiniciar la computadora.



Al finalizar el reinicio aparece la pantalla para iniciar sesión, donde se debe colocar el usuario y la clave que se escogió anteriormente y finalmente poder acceder al sistema.



La distribución Canaima por defecto tiene el usuario root deshabilitado por lo que para las tareas administrativas se utiliza sudo.

Por ejemplo:

- Si se desea instalar un paquete se debe escribir en consola `$ sudo aptitude install nombre_paquete`.
- Si se desea activar el usuario root se debe ejecutar la siguiente instrucción `$sudo -u root passwd`, se ingresa el password del usuario con el que se inició la sesión y luego se coloca una clave para el usuario root.
- Si más adelante se desea deshabilitar el usuario root, se puede hacer de la siguiente manera: `$sudo passwd -l root`

## UNIDAD II - SISTEMA X.Org

### EL SISTEMA X.Org.

X es el componente de los sistemas Unix encargado de mostrar la información gráfica, en particular, de dibujar los iconos, fondos y ventanas en las que se ejecutan las aplicaciones y es totalmente independiente del sistema operativo.

El sistema de ventanas X distribuye el procesamiento de aplicaciones especificando enlaces cliente-servidor. El servidor provee servicios para acceder a la pantalla, teclado y ratón (determina la resolución de la pantalla y la profundidad de color, mueve el cursor del ratón alrededor de la pantalla, entre otras acciones) mientras que los clientes son las aplicaciones que utilizan estos recursos para la interacción con el usuario. De este modo, mientras el servidor se ejecuta de manera local, las aplicaciones pueden ejecutarse remotamente desde otras máquinas, proporcionando así el concepto de transparencia de red.

X.Org es una implementación libre del sistema gráfico de ventanas X (también conocido como X11) que surgió como una bifurcación de Xfree86 después de un cambio de licencia que muchos consideran incompatible con la Licencia Pública General (GPL), esta ha sido adoptada por la mayoría de las distribuciones GNU/Linux.

### X-Windows

UNIX y GNU/Linux no incorporan la interfaz gráfica de usuario dentro del núcleo, en su lugar, es implementada por programas a nivel de usuario. Esto se aplica tanto a entornos gráficos como al modo texto. Esta disposición hace que el sistema sea más flexible, pero tiene la desventaja de que, al ser simple, implementar una interfaz de usuario diferente para cada programa, dificulta el aprendizaje del sistema.

El entorno gráfico principalmente utilizado con GNU/Linux se llama Sistema *X-Windows* (*X* para abreviar X11). *X* tampoco implementa por sí mismo una interfaz de usuario, sino solo un sistema de ventanas. Es decir, las herramientas bases con las cuales se puede construir una interfaz gráfica de usuario. Algunos administradores de ventanas populares son: FVWM, ICEWM, BLACKBOX Y WINDOW MAKER, METACITY. Existen también dos populares administradores de escritorios, KDE y GNOME.

## Modos VESA

VESA (Video Electronics Standards Association - *Asociación para estándares electrónicos y de video*) es una asociación internacional de fabricantes de electrónica. Fue fundada por NEC en los años 80 del siglo XX con el objetivo inicial de desarrollar pantallas de vídeo con una resolución común de 800x600 píxeles. Desde entonces, la VESA ha realizado otros estándares relacionados con funcionalidades de vídeo en periféricos de los IBM PC y compatibles, como conectores, BIOS ó características de la frecuencia, transmisión y sincronización de la imagen.

Los modos VESA más típicos son: hexadecimal y decimal.

### Hexadecimal

Colores	640×480	800×600	1024×768	1280×1024	1600×1200
256 (8 bits)	0×0301	0×0303	0×0305	0×0307	0×031C
32,768 (15 bits)	0×0310	0×0313	0×0316	0×0319	0×031D
65,536 (16 bits)	0×0311	0×0314	0×0317	0×031A	0×031E
16.8M (24 bits)	0×0312	0×0315	0×0318	0×031B	0×031F

### Decimal

Colores	640×480	800×600	1024×768	1280×1024	1600×1200
256 (8 bits)	769	771	773	775	796
32,768 (15 bits)	784	787	790	793	797
65,536 (16 bits)	85	788	791	794	798
16.8M (24 bits)	86	789	792	795	799

## Reconfigurar Servidor Gráfico X.org

Si por alguna razón después de realizar la instalación del sistema operativo se necesita configurar el

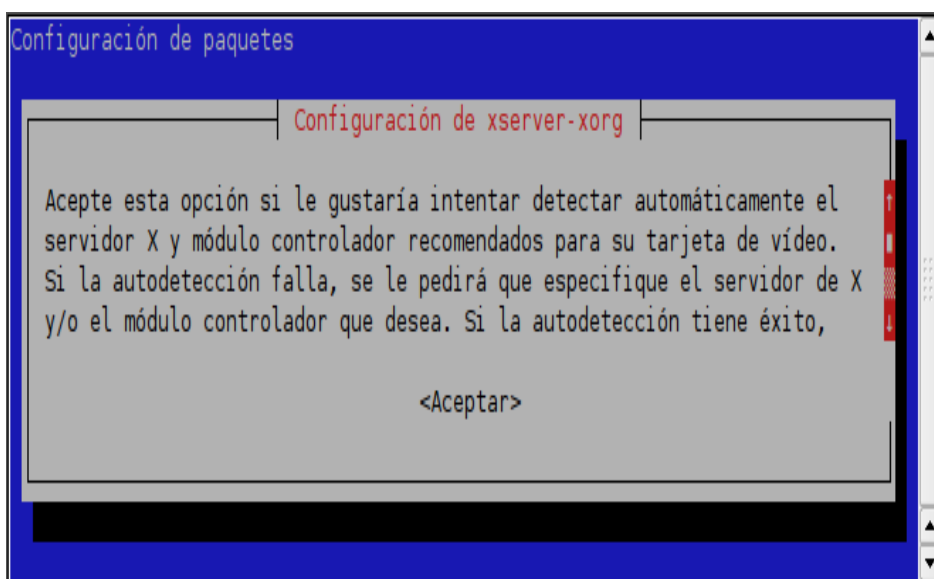
servidor gráfico de nuevo, por ejemplo, al no haber detectado en la instalación la resolución correcta del monitor, o en algún momento se cambia el monitor de la computadora y los parámetros que tienen configurados en el antiguo no funcionan con el nuevo, existe un script de configuración que ayuda a la reconfiguración del *Servidor X* sin necesidad de estar retocando a mano el fichero */etc/X11/xorg.conf*.

Para invocar el script se recomienda que se inicie sesión como usuario *root* en una consola de texto (CTRL+ALT+F1) y seguir los pasos a continuación:      *\$ login: root*

*\$ password:*

Antes de ejecutar el script, si ya se tiene configurado el servidor X, es recomendable que se realice una copia de seguridad del archivo *xorg.conf* de la siguiente manera: *# cp /etc/X11/xorg.conf /root*. Así se ha guardado una copia en el directorio de *root*. Para restaurarla se ejecutaría el siguiente comando: *# cp /root/xorg.conf /etc/X11/xorg.conf*.

Una vez hecha la copia de seguridad, se ejecuta el script de configuración: *# dpkg-reconfigure xserver-xorg*



## Las Secciones de *xorg.conf*

El archivo */etc/X11/xorg.conf* contiene la configuración de X.Org y está dividido en secciones. Cada sección empieza con la instrucción *<Section>*, seguido por el nombre de la sección entre comillas y

siempre termina con <EndSection>.

### ***Sección Monitor***

Define las propiedades del monitor. Las especificaciones del Sync Horizontal definen cuánto ancho de banda puede soportar el monitor y es especificado en kilohertz. Esto ayuda a identificar qué resolución es capaz de soportar el monitor. El Refresco Vertical dice cuantas veces por segundo el monitor puede refrescar las imágenes. Estas dos especificaciones pueden ser definidas en rango de valores que los monitores pueden soportar. Se recomienda revisar las especificaciones en el manual del monitor o buscar las especificaciones usando unas de las herramientas de configuración disponibles.

Otros parámetros que se encuentran en la sección Monitor son los de los modes. Se tiene dos maneras de especificarlos: el primero, usar la directriz ModeLine y especificar todos los números en una línea. El segundo, usar la subsección Mode, especificando los parámetros con el uso de marcados (tags). En ambas, éstos parámetros le comunican al Servidor X qué frecuencias y posicionamiento usar para cada resolución.

#### *Section "Monitor"*

*Identifier* "Failsafe Monitor"

*Vendorname* "AOC"

*Modelname* "AOC SPECTRUM 4V,4VA,4Vlr & 4VlrA, 4Vn, 4VnA"

*Horizsync* 30.0-50.0

*Vertrefresh* 50.0-100.0

*modeline* "800x600@56" 36.0 800 824 896 1024 600 601 603 625 +hsync  
+vsync

*modeline* "800x600@72" 50.0 800 856 976 1040 600 637 643 666 +hsync  
+vsync

*modeline* "800x600@60" 40.0 800 840 968 1056 600 601 605 628 +hsync  
+vsync

*modeline* "1024x768@60" 65.0 1024 1048 1184 1344 768 771 777 806 -vsync  
-hsync

*Gamma* 1.0

#### *EndSection*

## ***Sección Device***

Especifica los parámetros de la tarjeta de video. En la misma se puede especificar el chipset que el adaptador utiliza, cuota de RAM de video que tiene, la velocidad que puede usar y cualquier opción disponible para el driver asociado con el chipset utilizable. En la mayoría de los casos, no se necesita invocar éstos parámetros; ya que el servidor debe detectarlos.

### *Section "Device"*

*Identifier* "Failsafe Device"

*Boardname* "vesa"

*Busid* "PCI:1:0:0"

*Driver* "vesa"

*Screen* 0

### *EndSection*

Si por alguna razón el servidor no puede detectarlo correctamente, se pueden ingresar los parámetros correctos en esta sección. También se debe revisar la documentación del Xorg.

## ***Sección Screen***

Unifica toda la información necesaria desde las otras secciones. Se puede tener más de una sección de Device o Monitor en el archivo, pero sólo los listados en la sección Screen serán los utilizados, esta es la razón por la que cada sección incluye un identificador. De igual manera la sección screen especifica cuál módulo usar, la resolución y la intensidad del color.

### *Section "Screen"*

*Identifier* "Default Screen"

*Device* "Failsafe Device"

*Monitor* "Failsafe Monitor"

*Defaultdepth* 24

*SubSection "Display"*

*Depth* 24

*Virtual* 1024 768

*Modes* "640x480@85" "800x600@56" "640x480@75"



```

"800x600@72" "640x480@72" "800x600@75" "640x480@60"
"800x600@60" "832x624@75" "1024x768@60"
"1024x768@43"

```

*EndSubSection*

*EndSection*

### ***Sección Input Device***

Permite definir el protocolo que se va a usar para comunicarse con el mouse. Los protocolos del mouse incluyen PS/2, IMPS/2, Microsoft, y Logitech. Para todo lo que va desde el puerto PS/2, se usa */dev/psaux* como el dispositivo. Para los ratones seriales, */dev/ttySO* para el COM1, */dev/ttySl* para el COM2, y así sucesivamente. Muchas distribuciones permiten usar */dev/mouse* sin importar qué tipo de mouse se esté usando. En la sección Pointer, se puede especificar algunas opciones, como lo es emular el botón del medio haciendo uso del izquierdo y el derecho simultáneamente.

*Section "InputDevice"*

```

Identifier    "Configured Mouse"
Driver        "mouse"
Option        "CorePointer"
Option        "Device"      "/dev/input/mice"
Option        "Protocol"    "ImPS/2"
Option        "ZAxisMapping" "4 5"
Option        "Emulate3Buttons" "true"

```

*EndSection*

### ***Sección Files***

Informa al servidor de X dónde encontrar módulos de servidor, la base de datos de color RGB y archivos de tipografías. Esta opción es para usuarios avanzados. En la gran mayoría de los casos, se debería dejar activada.

## **SESIONES**

Una sesión es la duración de una conexión empleando una capa de sesión de un protocolo de red, o la duración de una conexión entre un usuario y un servidor, generalmente involucrando el intercambio de

múltiples paquetes de datos entre la computadora del usuario y el servidor. Típicamente es el tiempo que transcurre entre que un usuario se identifica en un sistema y, bien por falta de actividad, bien por desconexión voluntaria, el sistema deja de recordarle, la sesión le permite a un usuario, por ejemplo, estar conectado a los foros durante un tiempo determinado sin tener que volver a identificarse. A continuación se detalla los tipos de sesiones:

### ***Inicio de Sesiones desde Terminales***

El inicio de sesiones desde terminales (a través de líneas serie) y la consola (cuando no se está ejecutando X-Windows) es suministrado por el programa `getty`. `init` inicia una instancia independiente de `getty` por cada terminal en el que está permitido iniciar sesiones.

`Getty` lee el nombre de usuario y ejecuta el programa `login`, el cual se encarga de leer la `password`. Si el nombre de usuario y la `password` son correctas, `login` ejecuta el intérprete de comandos. Al finalizar el intérprete de comandos (en el caso en que, por ejemplo, el usuario finaliza su sesión; o cuando `login` finaliza debido a que no concuerdan el nombre de usuario y la `password`), `init` se entera de este suceso e inicia una nueva instancia de `getty`. El núcleo no tiene noción sobre los inicios de sesiones, esto es gestionado totalmente por los programas del sistema.

### ***Inicio de sesiones a través de la red***

Es tipo de sesión funciona de un modo un poco diferente al inicio de sesiones normales. Existe una línea serie física separada para cada terminal a través de la cual es posible iniciar sesión. Por cada persona iniciando una sesión a través de la red existe una conexión de red virtual, y puede haber cualquier número (no hay límite). Por lo tanto, no es posible ejecutar `getty` por separado por cada conexión virtual posible. Existen también varias maneras diferentes de iniciar una sesión a través de la red, las principales en redes TCP/IP son `Telnet` y `rlogin`.

Los inicios de sesión a través de la red tienen, en lugar de una cantidad enorme de `getty`'s, un servicio individual por tipo de inicio de sesión (`telnet` y `rlogin` tienen servicios separados) que *escucha* todos los intentos de inicio de sesión entrantes. Cuando el servicio advierte un intento de inicio de sesión, inicia una nueva instancia de si mismo para atender la petición individual; la instancia original continúa atenta a otros posibles intentos. La nueva instancia trabaja de manera similar a `getty`.

## UNIDAD III - EL SHELL

### USANDO EL SHELL

El intérprete de comandos es la interfaz entre el usuario y el sistema operativo; por esta razón, se le da el nombre en inglés *shell*, que significa caparazón. Por lo tanto, la shell actúa como un intermediario entre el sistema operativo y el usuario gracias a líneas de comando que este último introduce. Su función es la de leer la línea de comandos, interpretar su significado, llevar a cabo el comando y después arrojar el resultado por medio de las salidas.

La shell es un archivo ejecutable que debe interpretar los comandos, transmitirlos al sistema y arrojar el resultado. Existen varios shells. La más común es *sh* (llamada *Bourne shell*), *bash* (*Bourne again shell*), *csh* (*C Shell*), *Tcsh* (*Tenex C shell*), *ksh* (*Korn shell*) y *zsh* (*Zero shell*). Generalmente, sus nombres coinciden con el nombre del ejecutable.

Cada usuario tiene una shell predeterminada, la cual se activará cuando se abra un indicador del comando. La shell predeterminada se especifica en el archivo de configuración */etc/passwd* en el último campo de la línea que corresponde al usuario. Es posible cambiar de shell durante una sesión. Para esto, sólo se debe ejecutar el archivo correspondiente. Por ejemplo: */bin/bash*

#### Indicador del Sistema

La shell se inicia al leer su configuración completa (en un archivo del directorio */etc/*) y después al leer la configuración propia del usuario (en un archivo oculto cuyo nombre comienza con un punto y que se ubica en el directorio básico del usuario, es decir */home/user\_name/.configuration\_file*). A continuación, aparece el siguiente indicador llamado *prompt* en inglés:

*equipo:/directorio/actual\$*

De manera predeterminada, para la mayoría de las shells, el indicador consiste en el nombre del equipo, seguido de dos puntos (:), el directorio actual y después un carácter que indica el tipo de usuario conectado. Si el carácter es \$ especifica un usuario normal, si es # especifica un usuario administrador, llamado root.

## Línea de Comandos

Una línea de comandos es una cadena de caracteres formada por un comando que corresponde a un archivo ejecutable del sistema o, más bien, un comando de shell junto con argumentos opcionales (parámetros). Por ejemplo: *ls -al /home/jean-francois/*.

En el comando del ejemplo, *ls* es el nombre del comando, *-al* y */home/jean-francois/* son argumentos. Los argumentos que comienzan con *-* se denominan *opciones*. Por lo general, para cada comando, hay una cierta cantidad de opciones que se pueden detallar al introducir, por ejemplo el siguiente comando:  
*man nombre\_comando*

## Entrada-salida Estándar

Una vez que se ejecuta un comando, se crea un proceso. Este proceso abre tres flujos:

- 1) *stdin*, denominado *entrada estándar*, en cuyo caso el proceso lee los datos de entrada. De manera predeterminada, *stdin* se refiere al teclado. *stdin* se identifica con el número 0.
- 2) *stdout*, denominado *salida estándar*, en cuyo caso el proceso escribe los datos de salida. De manera predeterminada, *stdout* se refiere a la pantalla. *stdout* se identifica con el número 1.
- 3) *stderr*, denominado *error estándar*, en cuyo caso el proceso escribe los mensajes del error. De manera predeterminada, *stderr* se refiere a la pantalla. *stderr* se identifica con el número 2.

Por lo tanto, de manera predeterminada, cada vez que se ejecuta un programa, los datos se leen desde el teclado y el programa envía su salida y sus errores a la pantalla. Sin embargo, también es posible leer datos desde cualquier dispositivo de entrada, incluso desde un archivo, y enviar la salida a un dispositivo de visualización, un archivo, entre otros.

## Redirecciones

Como cualquier sistema Unix, GNU/Linux posee mecanismos que permiten redirigir la entrada-salida estándar a archivos. Por lo tanto, si se usa el carácter *>*, se puede redirigir la salida estándar de un comando que se encuentra a la izquierda a un archivo que se encuentra a la derecha. Por ejemplo:

```
ls -al /home/jf/ > toto.txt echo "Toto" > /etc/miarchivodeconfiguración
```

El propósito de la redirección *>* es el de crear un archivo nuevo. En el caso de que un archivo ya exista

con el mismo nombre, se debe eliminar. El siguiente comando simplemente crea un archivo vacío: `> nombre_archivo`.

El uso del carácter doble `>>` permite agregar la salida estándar al archivo, es decir, permite agregar la salida después del archivo sin eliminarlo. De manera similar, el carácter `<` indica una redirección de la entrada estándar. El siguiente comando envía el contenido del archivo *toto.txt* con el comando *cat*, cuyo único propósito es mostrar el contenido en la salida estándar (el ejemplo no es útil, pero es instructivo): `cat < toto.txt`.

Por último, el uso de la redirección `<<` permite la lectura, en la entrada estándar, hasta que se encuentre la cadena ubicada a la derecha. En el siguiente ejemplo, se lee la entrada estándar hasta que se encuentra la palabra *STOP*. Después, se muestra el resultado: `cat << STOP`.

## **Tuberías de Comunicación**

Las tuberías (en inglés *pipes* - literalmente *tuberías*) son mecanismos de comunicación específicos para todos los sistemas UNIX. Una tubería, simbolizada por una barra vertical (carácter `|`), permite asignar la salida estándar de un comando a la entrada estándar de otro, de la misma forma en que una tubería permite la comunicación entre la entrada estándar de un comando y la salida estándar de otro.

En el siguiente ejemplo, la salida estándar del comando *ls -al* se envía al programa *sort*, el cual debe extraer el resultado en orden alfabético. `ls -al | sort`. Esto permite conectar una cierta cantidad de comandos a través de sucesivas tuberías.

En el siguiente ejemplo, el comando muestra todos los archivos del directorio actual, selecciona las líneas que contienen la palabra "zip" (utilizando el comando *grep*) y cuenta la cantidad total de líneas: `ls -l | grep zip | wc -l`.

## **SHELL SCRIPTS**

Los shell scripts son programas escritos con comandos UNIX y son equivalentes a los batch de DOS aunque mucho más potentes, puesto que admiten ejecuciones en segundo plano y tienen un conjunto de expresiones mucho más amplia.

Una de las ventajas que presentan los shell scripts es que pueden ser portadas de una máquina UNIX a otra sin problemas, sin necesidad de retocar nada, salvo que se utilicen llamadas a programas muy

concretos específicos de una versión de UNIX, mientras que los programas compilados (desarrollados en C, Pascal, entre otros.) deben ser recompilados, pues el código se generará en función del microprocesador de cada máquina. Otra ventaja es la facilidad de lectura e interpretación.

El principal inconveniente que presentan respecto a los programas compilados es la lentitud de ejecución, que se puede paliar usando las utilidades *built-in* incluidas en el propio kernel en lugar de llamar a comandos externos que han de ser leídos de disco. Otro inconveniente es que el código resulta visible a cualquier usuario que lo pueda ejecutar.

En los shell scripts se deben añadir comentarios con el fin de facilitar la lectura del programa; los comentarios se insertan anteponiendo el carácter *#* al comentario, que se extenderá hasta el final de la línea. Estos deben colocarse en las cabeceras de los scripts indicando el nombre de archivo y lo que hace el script. Se colocan comentarios de documentación en diferentes partes del script para mejorar la comprensión y facilitar el mantenimiento. Un caso especial es el uso de *#* en la primera línea, seguido del carácter admiración y el path de la subshell, para indicar el intérprete con que se ejecutará el script. Por ejemplo: *#!/bin/sh*.

Es interesante saber que muchos comandos devuelven un valor después de ejecutarse, y que este valor indicará si la ejecución ha sido buena o si ha habido algún fallo y qué tipo de fallo se ha producido. Para conocer si un comando devuelve o no un valor y qué es lo que devuelve en cada caso se deberá consultar la documentación, pero por lo general en caso de una ejecución correcta devolverán el valor 0, y en caso de fallo otro número, positivo o negativo.

Para poder ejecutar un archivo de comandos es necesario que se tengan activados, al menos, los permisos de lectura y ejecución.

## **USANDO HISTORY**

La Instrucción history se utiliza para visualizar la lista de comandos previamente ejecutados, se invoca la utilidad history colocando simplemente: *# history*

### ***Variables de Entorno y Configuraciones***

Las variables de entorno y configuraciones son aquellas que tienen un significado propio para la shell o algún otro programa. Ciertos programas leen el contenido de las variables de entorno para

modificar su comportamiento, entre ellos la propia shell. Entre las variables de entorno más importantes se pueden citar:

- PATH, indica la ruta de búsqueda de programas ejecutables. Está constituida por una lista de directorios separados por dos puntos (:). El directorio actual, de forma predeterminada, no viene incluida en PATH.
- PS1, especifica el indicador del sistema. Lo habitual es que PS1 sea el símbolo \$ para usuarios normales y # para usuario root.
- PS2, especifica el indicador secundario del sistema. Aparece cuando no se ha completado una orden. LANG, especifica el lenguaje que se aplica al usuario; para español se utiliza *es*.
- LC\_ALL, contiene el idioma y se utiliza para usar los valores locales como mensajes del sistema, símbolo monetario, formato de fecha, formato de números decimales y otras características.
- TERM, almacena el tipo de terminal desde el que se está trabajando.
- EDITOR, especifica el editor por omisión del sistema. Lo habitual en los sistema Unix es que el editor por omisión sea *vi*.
- DISPLAY, especifica qué equipo muestra la salida que se efectúa en modo gráfico. Ese equipo deberá tener un servidor gráfico.
- LD\_LIBRARY\_PATH, se utiliza para definir rutas alternativas de búsqueda para bibliotecas de funciones del sistema.
- PWD, contiene el directorio de trabajo efectivo.
- Dmidecode, información sobre las características del hardware del sistema.

- Last, información sobre los últimos usuarios que han usado el sistema.

Con la orden *environment* se puede comprobar el valor de las variables de entorno del sistema. Para modificarlas basta asignarle un nuevo valor.

## PROCESOS

El shell utiliza el kernel para la ejecución de procesos, los cuales quedan bajo su control. Es posible definir un *proceso como un programa en ejecución*. Ya que UNIX es multitarea, utiliza una serie de métodos de tiempo compartido en los cuales parece que hay varios programas ejecutándose a la vez, cuando en realidad lo que hay son intervalos de tiempo cedidos a cada uno de ellos según un complejo esquema de prioridades.

### *Propiedades de los Procesos*

Básicamente, un proceso tiene las siguientes propiedades:

- Un número identificador, (Process ID o PID), identificador de proceso, es necesario para referirse a un proceso en concreto de los varios que se encuentran en ejecución.
- Un PPID (Identificador del Proceso Padre), es el número que indica qué proceso creó al proceso en cuestión.

### *Estado de los Procesos*

Hay momentos en los que un proceso sigue existiendo en el sistema, pero en realidad no están realizando algo, quizás porque pueden estar esperando a que una *señal* le sea enviada para volverse activo, o a un usuario le puede interesar detenerlo o pausarlo bajo determinadas circunstancias. Los estados más importantes son *dormido (S)*, y en *ejecución (R)*.

### *Procesos en Ejecución y sus Propiedades*

Para empezar, se pueden ver las tareas y las subtareas en una estructura anidada mediante el comando *pstree* es decir, permite visualizar un árbol de procesos. Asimismo, *pstree -p* muestra entre paréntesis el número identificador (PID) de los procesos, algo muy importante cuando se quiere pasar de actuar de forma pasiva a interactuar con los procesos, cosa que normalmente se hace señalando sus PIDs. Aunque dicha información estructurada resulta interesante, existen dos comandos muy conocidos que



muestran una cantidad ingente de información sobre los procesos, estos son:

**Comando *ps***, muestra una lista de los procesos en ejecución. Las opciones más habituales son: *ps u* → que muestra los procesos que pertenecen al usuario actual, *ps aux* → muestra información detallada de todos los procesos en ejecución. Algunos de los campos más importantes mostrados por *ps* son:

- USER - usuario dueño del proceso.
- PID - número identificador del proceso.
- %CPU - porcentaje de uso del microprocesador por parte de este proceso.
- %MEM - porcentaje de la memoria principal usada por el proceso.
- VSZ - tamaño virtual del proceso (lo que ocuparía en la memoria principal si todo él estuviera cargado, pero en la práctica en la memoria principal sólo se mantiene la parte que necesita procesarse en el momento).
- RSS - tamaño del proceso en la memoria principal del sistema (generalmente son KBytes, cuando no lo sea, se indicará con una M detrás del tamaño).
- TTY - número de terminal (consola) desde el que el proceso fue lanzado. Si no aparece, probablemente se ejecutó durante el arranque del sistema.
- STAT - estado del proceso.
- START - cuándo fue iniciado el proceso.
- TIME - el tiempo de CPU (procesador) que ha usado el proceso.
- COMMAND - el comando que inició el proceso.

**Comando *top***, es la versión interactiva de *ps*, y tiene algunas utilidades interesantes añadidas. Si se ejecuta en una terminal y sin opciones, aparecerá arriba información del sistema: usuarios, hora, información del tiempo de funcionamiento de la máquina, número de procesos, uso de CPU, uso de memoria y uso del swap y a continuación muestra una lista de procesos similar a la que se muestra con *ps*, la diferencia entre ambos radica en que ésta se actualiza periódicamente, permitiéndo ver la evolución del estado de los procesos.

Con *top*, se tiene dos posibilidades de especificar opciones, bien en línea de comandos en el shell, o

bien interactivamente (mientras está en ejecución y sin salir de él). La página del manual de `top` es también muy buena, con descripciones detalladas de los campos y de las opciones, tanto de línea de comandos como interactivas.

### ***Tareas de Bash - Programas en Primer y Segundo Plano.***

El control de tareas es una interesante característica que `bash` ofrece. Así, cada programa o tubería que se ejecute tiene asignado un número de tarea por parte de `bash` (diferente al PID). Los programas que se pueden ejecutar desde una línea de comandos pueden estar en primer o segundo plano, y de hecho pueden pasar de un estado a otro.

Una consola concreta está bloqueada cuando un proceso está en primer plano, mientras que si está en segundo plano la consola está libre y se puede teclear comandos en ella. Esto está más relacionado con la terminal en sí que con el concepto de proceso que se ha aprendido.

Normalmente al ejecutar un comando no se recupera el prompt hasta que éste no termina. La necesidad que surge es poder usar esa terminal sin abrir otra nueva a la vez que el comando continúe haciendo lo que tenga que hacer.

En `bash`, se colocan las tareas en segundo plano añadiendo un `&` al final del comando. El comando se ejecuta mientras la terminal queda libre. Por ejemplo:

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

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

Teniendo una tarea en segundo plano, arrojará su salida a la terminal actual a pesar de todo. En estos casos se pueden usar las redirecciones para evitarlo. Para recuperar una tarea en segundo plano a primer plano, se puede usar `fg`, seguido de `%N`, donde `N` es el número de tarea que se quiere recuperar, por ejemplo `fg %2` traería a primer plano la segunda tarea.

Con una tarea en primer plano, se puede usar `Ctrl+Z` y se detendrá. Ahora se puede elegir entre continuar con dicha tarea en primer plano, o mandarla a segundo plano. Para continuarla en primer plano, se teclea `fg %1`. Para seguirla en segundo plano, se usa `bg %1`, suponiendo que es la tarea número 1.

Un comando ejecutándose en primer plano puede ser cancelado con `Ctrl+C`, lo que a su vez hará desaparecer al proceso resultante de la tabla de procesos del kernel. El comando *kill -SEÑAL PID* es el que se usa para enviar señales a los procesos. Se puede indicar varios PIDs para que reciban esa señal en un solo comando separándolos con espacios al final. Son equivalentes: `# kill -SIGKILL 1283`

`# kill -9 1283`

Para un usuario, observar en cada momento qué PID tiene el proceso originado por un comando puede ser incómodo. Para eso existe el comando *killall -SEÑAL nombre\_comando*, donde "nombre\_comando" es el nombre del comando que dió lugar al proceso en cuestión.

### ***Prioridad de los Procesos.***

Lo más común es tener muchos procesos a la vez ejecutándose en una computadora. Pero no todos son igual de importantes. Por ejemplo, si se está grabando un CD, este debería ser un proceso más importante que el resto, porque si el disco duro no envía los datos a la velocidad suficiente a la grabadora, se pierde el disco grabable.

En dicha situación se le dice al kernel: *si el procesador no puede con todo, lo último que debes retrasar es la grabación del CD*. Se puede notar entonces, que el resto de las aplicaciones van más lentas, pero eso no garantiza que la grabación no va a ser interrumpida. Esto se puede resolver ejecutando desde el principio el comando interesado usando *nice*, o bien, conseguir su PID si ya se está ejecutando y usar *renice* para cambiar su prioridad. Por ejemplo: `$ nice -n prioridad comando`

`-$ renice prioridad pid_proceso`

Donde:

- *prioridad*, es un valor que va desde -20 a +20 (con el signo incluido). -20 es la prioridad más alta (al contrario de lo que se puede pensar) y +20 la más baja. Sólo root puede establecer una prioridad negativa a un proceso, los usuarios como máximo pueden colocar un proceso en prioridad 0.
- *comando*, es el comando que se quiere ejecutar (sólo aplicable con nice), el mismo comando que se ejecutaría normalmente con todos sus parámetros y opciones.
- *proceso* (sólo aplicable con renice) cambia la prioridad del proceso cuyo PID es PID\_PROCESO al nuevo valor indicado.

## COMANDOS

Un comando es una instrucción o mandato que el usuario proporciona a un sistema informático, desde la línea de comandos (como una shell) o desde una llamada de programación. Puede ser interno (contenido en el propio intérprete) o externo (contenido en un archivo ejecutable). Suele admitir parámetros (argumentos) de entrada, lo que permite modificar el comportamiento predeterminado del comando. Suelen indicarse tras una barra / en sistemas operativos DOS ó un guión simple - ó doble -- en sistemas operativos Unix. A continuación algunos comandos de gran utilidad.

### *Comando ps*

Cuando el shell lanza un programa, se crea un nuevo proceso y se le asigna un número entero (PID) entre 1 y el 30.000, del cual se tiene la seguridad que va a ser unívoco mientras dure la sesión. Se puede verificar ejecutando el comando *ps*, el cual muestra los procesos activos que se tienen asociados a la terminal.

Un proceso que crea a otro se lo denomina proceso padre. El nuevo proceso, en este ámbito se le denomina proceso hijo. Este hereda casi la totalidad del entorno de su padre (variables, entre otras), pero sólo puede modificar su entorno, y no el del padre. La mayoría de las veces, un proceso padre se queda en espera de que el hijo termine, esto es lo que sucede cuando se ejecuta un comando, el proceso padre es el shell, que lanza un proceso hijo (el comando). Cuando este comando acaba, el padre vuelve a tomar el control, y recibe un número entero donde recoge el código de retorno del hijo (0 = terminación sin errores, otro valor = aquí ha pasado algo).

Cada proceso posee también un número de *grupo de procesos*. Los miembros de un grupo de procesos (procesos cuyo PGID es igual al PGID de la terminal en curso) reciben señales generadas por el teclado como *SIGINT*. Se dice que estos procesos están en *primer plano*. Los procesos en *segundo plano* son aquéllos cuyo PGID difiere del de la terminal; tales procesos son inmunes a señales generadas desde el teclado. Sólo los procesos en primer plano tienen permitido leer o escribir en la terminal. A los procesos en segundo plano que intenten leer de (o escribir en) la terminal, el controlador de terminal les manda una señal *SIGTTIN* (*SIGTTOU*) que, a menos que sea capturada, suspende el proceso.

Es posible utilizar algunas variantes del comando *ps* para ver qué procesos se tienen en el equipo:

- *ps -e* : de todas las sesiones.
- *ps -f* : full listing: da los números del PID, del Ppid (padre), uso del procesador y tiempo de comienzo.
- *ps -j*: da el PGID (número de grupo de los procesos - coincide normalmente con el padre de todos ellos).

### ***comando kill***

Este comando sirve para matar o anular procesos indeseados. Se debe tener en cuenta que cada proceso lleva su usuario y por tanto solo él (o el superusuario) pueden matarlo. Normalmente, si los programas que componen el grupo de procesos son civilizados, al morir el padre mueren todos ellos siempre y cuando el padre haya sido señalado adecuadamente. Para ello, se emplea el comando *\$kill <número\_señal> PID*, siendo *PID* el número del proceso o del grupo de procesos. Los números de señales (*número\_señal*) utilizados con más frecuencia son:

- -15: TERM o terminación, se manda para que el proceso cancele ordenadamente todos sus recursos y termine.
- -1: corte.
- -2: interrupción.
- -3: quit.
- -5: hangup.

- -9: kill, la más enérgica de todas pero no permite que los procesos mueran ordenadamente. El proceso que la recibe finaliza inmediatamente.

### **Comando ls**

Lista el contenido del directorio en el que se encuentra el usuario. Si se coloca sólo *ls* se obtiene una lista con el nombre de los archivos; si se quiere obtener más información sobre esos archivos se utilizan las opciones del comando que se muestran a continuación:

- -a → lista los archivos invisibles es decir, los que empiezan por punto.
- -s → muestra el tamaño del archivo en kilobytes (1024 bytes). Precede al nombre de cada archivo.
- -F → añade un slash ( / ) a los directorios y un asterisco ( \* ) a los archivos ordinarios ejecutables.

Estas opciones se pueden combinar para obtener la información que se quiera al mismo tiempo; por ejemplo, *ls -sF*, dará la lista de los archivos en la que el nombre de cada archivo va precedido por su tamaño (en kilobytes) y va seguido de un slash (/) en el caso de que sea un directorio o de un asterisco (\*) en el caso de que sea un archivo ejecutable.

De igual manera, se puede listar el contenido de un directorio diferente al que está el usuario, en ese caso solo se debe especificar el path correspondiente a continuación de las opciones requeridas. Por ejemplo: *ls /usr/share/doc/*

### **Comando cat**

El comando cat (concatenate) se utiliza para visualizar por pantalla el contenido de uno o más archivos. Cuando se especifica más de un archivo, cat los edita uno detrás de otro. La sintaxis del comando es: *\$ cat [-ns] archivo(s)*, donde:

- -n → numera las líneas.
- -s → elimina las líneas en blanco.
- archivo(s) → nombre o nombres de los archivos que se van a editar.

El comando cat no pagina, entonces se utiliza: CTRL-S para parar la pantalla y CTRL-Q para continuar

con la edición. Si el sistema es demasiado rápido se puede utilizar el comando *more* (el cual se explicará más adelante). Por ejemplo: *cat archivo | more*.

El comando *cat* permite también concatenar archivos; para ello se ejecutaría el siguiente comando : *\$ cat archivo1 archivo2 ... > archivo n* , que permite unir los archivos *archivo1*, *archivo2*, ... y lo almacena en el *archivo n*.

Si se utiliza por equivocación el comando *cat* sin ningún argumento, intenta leer de la pantalla, por lo que no sale el prompt del sistema (se queda como colgada); entonces hay que pulsar **CTRL-C** para salir.

### ***Comando grep***

Busca una cadena de caracteres en uno o más archivos y lista todas las líneas que la contienen. La sintaxis del comando es: *\$ grep [- v l i w n r ] cadena archivo(s)*, donde:

- *-v* → lista las líneas que no contienen la cadena de caracteres.
- *-l* → lista el nombre del archivo que contiene la cadena de caracteres.
- *-i* → ignora la diferencia entre letras mayúsculas y minúsculas.
- *-w* → se utiliza cuando la cadena de caracteres es una única palabra.
- *-n* → muestra el número de la línea en la que se encuentra la cadena de caracteres.
- *-r* → en caso que sea un directorio, para buscar en todos los que cuelgan de él.
- *cadena* → cadena de caracteres que se quiere buscar.
- *archivo(s)* → nombre o nombres de los archivos en los que se quiere buscar la cadena de caracteres especificada.

### ***Comando more***

Se utiliza para editar archivos por la pantalla; la principal diferencia con *cat* es que se puede controlar el número de líneas que aparecen en la pantalla utilizando las teclas siguientes:

- Barra Espaciadora → se avanza una página.
- Tecla Return → se avanza una línea.

- Tecla DEL ó Q → se sale de la edición.

La sintaxis del comando *more* es: *\$ more [-cd] [+número de líneas] [+/path] archivo(s)* donde:

- -c → edita pantalla a pantalla.
- -d → número de líneas que se van a editar.
- +número de líneas → número de la línea a partir de la cual se va a editar.
- +/path → path correspondiente al archivo que se va a editar.
- archivo(s) → nombre o nombres de los archivos que se van a editar.

Por ejemplo: *more -c10 +25 +/usr/share/doc/gedit/README*, muestra 10 líneas, empezando por la 25, del archivo llamado README que se encuentra en el directorio */usr/share/doc/gedit*.

El comando *more* se puede usar con otros comandos para paginar la salida por pantalla.

### ***Comando cp***

Se utiliza para copiar archivos. Su sintaxis es: *cp [-i] archivo\_entrada archivo\_destino*, donde:

- -i → origina que el comando requiera una confirmación, en el caso de que el archivo destino ya exista es decir, pregunta si se desea hacer la copia.
- -r → para copiar un directorio completo.
- archivo\_entrada → nombre del archivo que se va a copiar.
- archivo\_destino → nombre del archivo en el que se va a copiar el contenido del archivo de entrada

### ***Comando du***

El comando *du* informa al usuario de la cantidad de almacenamiento utilizado por los archivos especificados, posee varias opciones, su sintaxis es la siguiente: *du [opciones][archivo]*. Sus opciones más significativas son:

- -s → muestra únicamente los tamaños de los archivos especificados en la línea de comandos.
- -h → muestra los tamaños de archivo en un formato más legible.



- `-c` → muestra en pantalla el espacio total ocupado por los archivos especificados.
- `-x` → omite en el conteo aquellos directorios que pertenezcan a otro sistema de archivos.

### ***Comando df***

Informa sobre la ocupación de disco que realiza el sistema. Por defecto, esta utilidad muestra el tamaño de las particiones en bloques de 1 kilobyte y el tamaño del espacio libre en kilobytes. Para ver esta información en megabytes y gigabytes, se utiliza el comando `$ df -h`.

### ***Comando fdisk***

Es una herramienta que permite crear particiones de disco y escribe la tabla de particiones en el sector 0. Cuando se utiliza sin parámetros presenta un menú de opciones con las que se puede interactuar. Algunas opciones de utilidad son:

- `-l` → lista las tablas de particiones.
- `-v` → muestra la versión de fdisk.

### ***Comando rm***

Se utiliza para borrar archivos. La sintaxis de este comando es: `$ rm [-i] archivo(s)`, donde:

- `-i` → origina que el comando requiera confirmación para ejecutarse.
- `archivo(s)` → nombre o nombres de los archivos que se van a borrar.

### ***Comando mv***

Se utiliza para renombrar archivos es decir, el contenido del archivo no cambia, sólo cambia el nombre o para mover archivos entre directorios. La sintaxis del comando es:

`$ mv [-i] archivo_entrada archivo_destino`

Donde:

- `-i` → origina que el comando requiera una confirmación, en el caso de que el archivo destino ya exista; es decir, pregunta si se desea hacer la copia.
- `archivo_entrada` → nombre del archivo que se va a renombrar.

- `archivo_destino` → nombre del archivo en el que se va a copiar el contenido del archivo de entrada.

### ***Comando pwd***

Muestra la ruta completa del directorio en el que está el usuario. Sintaxis: *\$pwd*.

### ***Comando cd***

Permite cambiar el directorio de trabajo (o directorio actual). Ejemplos:

- `cd /directorio` → cambia al directorio utilizando la ruta absoluta.
- `cd directorio` → cambia al directorio utilizando una ruta relativa al directorio de trabajo actual.
- `cd ~luis` → cambia al directorio de trabajo (home) del usuario Luis.

Otras opciones de navegación entre directorios son *cd .* → que se refiere al directorio actual y *cd ..* se refiere al directorio inmediatamente anterior (superior) al de trabajo actual.

### ***Comando mkdir***

Permite crear directorios. Sintaxis: *\$ mkdir nombre\_directorio*.

### ***Comando rmdir***

Permite borrar directorios, su sintaxis *\$ rmdir nombre\_directorio*. Si el directorio no está vacío se puede ejecutar *\$ rm -r nombre\_directorio*.

### ***Comando gzip***

gzip es el compresor por excelencia en cualquier sistema Unix. Sus operaciones básicas son:

- Comprimir un archivo: `gzip [-n] archivo`, siendo n un número del 1 al 9 donde 1 es más rápido y 9 más comprimido.
- Descomprimir un archivo: `gzip -d archivo.gz`

También existe otro compresor `bzip2`, que a pesar de ser más lento es bastante más eficiente. La sintaxis es prácticamente la misma, solo que el sufijo del archivo es `.bz2`.

En cualquier caso, la forma estándar de juntar y comprimir varios archivos consiste en hacer un `.tar`

con todos ellos y luego comprimir el archivo con gzip

### ***Comando tar***

tar es la forma estándar de hacer un volumen de archivos (un archivo que contiene varios archivos). Hay que notar que *tar* no comprime el volumen resultante. Somos nosotros los que elegimos el algoritmo de compresión mediante otro programa (normalmente gzip). Sintaxis básica: *tar [OPERACIONES Y OPCIONES] archivos\_involucrados* donde las operaciones básicas son:

- Creación de un volumen: *tar -cf archivo.tar archivo\_o\_dir1 [archivo\_o\_dir2] ...*
- Añadir archivos a un volumen: *tar -rf archivo.tar archivo\_o\_dir1 [archivo\_o\_dir2] ...*
- Extraer archivos de un volumen: *tar -xf archivo.tar [archivo\_o\_dir1] [archivo\_o\_dir2] ...*
- Listar los archivos contenidos: *tar -tf archivo.tar*

### ***Comando gunzip***

Es muy común encontrarse en internet este tipo de formato y es tan fácil de descomprimir con solo colocar en el interprete de comandos: *gunzip nombre\_del\_archivo.gz*

### ***Comando fsck***

En ciertas ocasiones es necesario verificar la integridad del sistema de archivos y corregir los posibles errores que hubiese. Esta acción la realiza la orden *fsck*. Para verificar un sistema de archivos es aconsejable hacerlo mientras este está desmontado. Es una forma de evitar riesgos innecesarios. *#fsck [-opciones] /dev/hdXXX*, donde opciones:

- *-a* → confirmar automáticamente. ¡Usar con cuidado!
- *-c* → comprobar bloques malos en el disco.
- *-v* → verbose, despliega más información.
- *-r* → reparar pidiendo confirmación (Modo interactivo).
- *-y* → asume respuesta de "yes" siempre (se corre un riesgo).
- *-f* → forzar el chequeo aunque todo parezca en orden.

## ***Comando ln***

En Linux pueden definirse enlaces a elementos del sistema de archivos para poder acceder a ellos desde distintos lugares en la jerarquía. Un enlace no es más que un nombre que apunta a un determinado recurso del sistema de archivos, sea físico o lógico. Debido a esto se clasifican en dos tipos:

- *Fuertes o duros*: son aquellos que no se diferencian en nada del archivo original. Realmente un archivo existe físicamente (ocupa una zona en algún dispositivo de almacenamiento) y su nombre no es más que un enlace fuerte a él. Si se creara otro enlace fuerte, solo se estaría apuntando a la misma zona física a través de otro nombre. De esta forma se obtendrían dos o más copias lógicas de un archivo, pero solo habría una copia física. De aquí se deduce que un archivo no desaparece físicamente hasta que no se borren todos los enlaces fuertes que apunten a él. Los enlaces duros a un archivo determinado se almacenan en la estructura del i-nodo que lo representa.

Si modificamos un archivo, los mismos cambios aparecerán en cualquier enlace fuerte. Imaginemos que queremos compartir en red un archivo situado en nuestro directorio de trabajo, pero que no queremos compartir todo el directorio. La solución más simple consiste en crear un enlace fuerte desde dentro de un directorio que sí queramos compartir.

- *Simbólicos o débiles*: son apuntadores al nombre del archivo, no a su contenido. Si desaparece el archivo original (todos los enlaces duros a este) los enlaces simbólicos correspondientes quedan inconsistentes.

No se pueden crear enlaces fuertes a directorios, ni a archivos en particiones distintas. Para crear enlaces se emplea el comando *ln*.

Sintaxis: *ln [opciones] archivo\_o\_directorio [nombre\_del\_enlace]*, donde opción:

- *-s* → se utiliza para crear enlaces simbólicos en lugar de fuertes (como es por defecto).

Nota: el nombre del archivo o el directorio al que queremos hacer el enlace debe escribirse con todo el path, es decir, *"/ruta\_al\_archivo/archivo"*.

Ejemplos:

- `ln -s /home/pepe/public_html/raiz.html index.html` Hace que al acceder al archivo "index.html" nos envíe al archivo "raiz.html".
- `ln -s /usr/bin /binarios` Si entramos en el directorio "/binarios" nos enviará al directorio "/usr/bin"

## UNIDAD IV - GESTIÓN DE USUARIOS Y GRUPOS

### GESTIÓN DE USUARIOS

Linux es un sistema multiusuario y permite que varios usuarios puedan acceder, incluso simultáneamente. Cada usuario podrá tener su configuración y sus archivos independientes.

Los grupos permiten asignar permisos de archivos y directorios a muchos usuarios de una vez.

A un grupo pueden pertenecer varios usuarios y un usuario puede pertenecer a varios grupos. Un usuario tiene asignado un grupo principal o por defecto.

El Superusuario dentro del entorno de GNU/Linux, es aquel usuario que posee todos los privilegios dentro del sistema, es capaz de realizar cualquier operación dentro del sistema, es equivalente al usuario administrador dentro de los sistemas Microsoft Windows. Además de entrar en el login del sistema como root, hay dos formas para ampliar los privilegios de un usuario y adquirir los de root. Los dos programas para hacer esto son `su` y `sudo`.

El comando `su` hace que un usuario que se haya identificado con su propia cuenta pueda cambiar su uid al de root. Por supuesto debe saber el password del root.

El comando `sudo`, en este caso no es necesario que el usuario conozca la contraseña de root.

Este programa permite que un usuario pueda ejecutar determinados comandos con privilegios de root. Estos usuarios y los comandos permitidos para él deben de estar en el archivo `/etc/sudoers`.

Por ejemplo para que el usuario carlos pueda hacer un shutdown del sistema debe haber una entrada en el archivo `sudoers` como: `carlos /sbin/shutdown -[rh] now`.

Comandos más utilizados:

- *Comando who*: información sobre los usuarios que usan el sistema en este momento.
- *Comando finger*: información sobre el usuario *usuario*.
- *Comando adduser*: registra y crea una cuenta de *usuario*.

En ese momento, no sólo se creará la cuenta del usuario sino también su directorio de trabajo, un nuevo grupo de trabajo que se llamará igual que el usuario y añadirá una serie de archivos de configuración al directorio de trabajo del nuevo usuario:

```
root@cila:/home# adduser luis
Adding user luis...
Adding new group luis (1000).
Adding new user luis (1000) with group luis
Creating home directory /home/luis
Copying files from /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for luis
Enter the new value, or press return for the default
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct?
[y/n] y
```

En ese momento, el usuario ya puede trabajar en el sistema. Otro comando utilizado es *deluser*, el cual borra la cuenta de usuario *usuario*. Este comando no elimina automáticamente el directorio de trabajo del usuario.

```
root@cila:/home# deluser luis

Removing user luis...

done.
```

Una vez realizado este proceso, es responsabilidad del administrador decidir si elimina el directorio de

trabajo del antiguo usuario.

*Comando passwd:* cambia la clave de acceso para el usuario actual. *root* puede cambiar la clave de cualquier usuario con *passwd usuario*. # passwd victor.

La base de datos básica de usuarios en un sistema Unix es un archivo de texto */etc/passwd* (llamado el archivo de contraseñas), que lista todos los nombres de usuarios validos y su información asociada.

El archivo tiene una línea por usuario, y es dividido en siete colon-delimited campos:

- nombre de usuario.
- contraseña, de modo encriptado.
- Identificación (Id) de numero de usuario.
- Identificación (Id) de numero de grupo
- Nombre completo u otra información descriptiva de la cuenta.
- Directorio Inicio (directorio principal del usuario).
- Interprete de comandos (programa a ejecutar al ingresar al sistema).

Cualquier usuario del sistema puede leer el archivo de contraseñas, para por ejemplo conocer el nombre de otro usuario del mismo. Esto significa que la contraseña (el segundo campo) esta también disponible para todos. El archivo de contraseñas encripta las contraseñas, así que en teoría no hay problema, pero dicho encriptado puede ser quebrado, sobre todo si dicha contraseña es débil. Por lo tanto no es buena idea tener las contraseñas en el archivo de contraseñas.

Muchos sistemas GNU/Linux tienen contraseñas sombra. Esto es una alternativa en la manera de almacenar las contraseñas: las claves encriptadas se guardan en un archivo separado */etc/shadow* que solo puede ser leído por el administrador del sistema. Así el archivo */etc/passwd* solo contiene un marcador especial en ese segundo campo. Cualquier programa que necesite verificar un usuario o uid, pueden también acceder al archivo *shadow/sombra*. Significa también que programas normales que solo usan otros campos del archivo de contraseñas, no pueden acceder a las contraseñas. Paralelamente también existe */etc/gshadow* para cierta información según grupos.



## ***Crear un usuario manualmente***

Para crear una nueva cuenta a mano, sigue estos pasos:

- Editar `/etc/passwd` con `vipw` y agregar una nueva línea por cada nueva cuenta. Teniendo cuidado con la sintaxis. No se debe editar directamente con un editor, se debe usar `vipw` que bloquea el archivo, así otros comandos no tratarán de actualizarlo al mismo tiempo. Se debería hacer que el campo de la contraseña sea ``*'`, de esta forma es imposible ingresar al sistema.
- Similarmente, edite `/etc/group` con `vigr`, si necesita crear también un grupo.
- Cree el directorio Inicio del usuario con el comando `mkdir`.
- Copie los archivos de `/etc/skel` al nuevo directorio creado.
- Corrija la pertenencia del dueño y permisos con los comandos `chown` y `chmod`. La opción `-R` es muy útil. Los permisos correctos varían un poco de un sitio a otro, pero generalmente los siguientes comandos harán lo correcto:

*`cd /home/nuevo-nombre-de-usuario`*

*`chown -R nombre-de-usuario.group .`*

*`chmod -R go=u,go-w . chmod go= .`*

*Asigne una contraseña con el comando `passwd`*

Después de asignar la contraseña del usuario en el último paso, la cuenta funcionará. No debería configurar esto hasta que todo lo demás esté hecho, de otra manera el usuario puede inadvertidamente ingresar al sistema mientras copia los archivos de configuración de su entorno de trabajo.

A veces es necesario crear cuentas falsas que no son usadas por personas. Por ejemplo, para configurar un servidor FTP anónimo (así cualquiera podrá acceder a los archivos por él, sin tener que conseguir una cuenta de usuario en el sistema primero) podría crear una cuenta llamada "ftp". En esos casos, usualmente no es necesario asignar una contraseña (el último paso de arriba). Verdaderamente, es mejor no hacerlo, para que nadie puede usar la cuenta, a menos que primero sea root/cuenta administrador, y así convertirse en cualquier usuario.

## GESTIÓN DE GRUPOS

Cuando creamos un usuario, siempre lo vamos a incluir en algún grupo de trabajo, ya sea el suyo propio o bien, en uno común. Los comandos utilizados para el manejo de grupos son:

**Comando *addgroup***, crea el grupo *grupo*.

La forma de hacerlo es: `root@cila:/home# addgroup usuarios`

Adding group usuarios (105)...

Done.

El número 105 nos indica que ése es el identificador numérico que se le asigna al nuevo grupo en el momento de su creación.

**Comando *delgroup***, borra el grupo. Sintaxis: `# delgroup nombre_grupo`. De forma similar, la eliminación de un grupo se hace de esta forma:

`root@cila:/home# delgroup usuarios`

Removing group usuarios...

Done.

¿Qué puede pasar si tratamos de eliminar un grupo inexistente? El sistema nos avisará con el siguiente mensaje: `root@cila:/home# delgroup usuarios`

`/usr/sbin/delgroup: `usuarios' does not exist.`

### ***Añadiendo y Eliminando usuarios de los grupos.***

Para añadir un usuario pepe a un grupo usuarios haremos:

`root@cila:/home# adduser carla usuarios`

Adding user pepe to group usuarios...

Done.

Para eliminarlo de ese grupo:

`root@cila:/home# deluser pepe usuarios`

Removing user pepe from group usuarios...

done.

## PERMISOS DE ARCHIVOS

El sistema UNIX posee un medio sencillo para controlar quién puede acceder o no a sus archivos.

Existen tres clases diferentes de usuarios de un archivo y tres modos diferentes de acceso al archivo.

Entre ellos se encuentran:

- Propietario: usuario que ha creado el archivo. El propietario tiene capacidad de controlar quien puede acceder al archivo.
- Grupo: grupo de usuarios, normalmente relacionados por un departamento o función. Un usuario de este tipo puede acceder al archivo, pero no puede cambiar quien puede acceder al archivo.
- Otros: cualquier otro usuario del sistema. Estos usuarios pueden únicamente acceder al archivo si tienen permiso para ello.

<b>rwX</b>	<b>rwX</b>	<b>rwX</b>
Permisos para el usuario propietario	Permisos para el grupo de usuarios	Permisos para otros usuarios

Para cada una de las tres clases de usuarios existen 3 modos de acceso diferentes, como se muestran en la siguiente tabla:

<u><b>MODO</b></u>	<u><b>ARCHIVO</b></u>	<u><b>DIRECTORIO</b></u>
Lectura(r)	Examinar el contenido.	Listar los archivos contenidos en él.
Escritura(w)	Cambiar el contenido.	Crear y borrar archivos.
Ejecución(x)	Ejecutarlo como un comando.	Buscar en el directorio.

## PERMISOS: SUID y SGID

Hablaremos brevemente de dos permisos especiales: SGID y SUID. En cuanto a estos permisos, decir que los programas con este tipo de permisos especiales respetan los permisos del propietario aun

cuando sean usados por otro usuarios. Es decir, que si un usuario ejecuta un archivo con SUID y el propietario es root, tendrá privilegios de root en ese archivo con los peligros que conlleva.

Los archivos con SUID podemos buscarlo introduciendo este simple comando en una terminal: `# find / -perm +4000 (+2000 para SGID)`.

Con los permisos es muy utilizado el comando `chmod` (change mode) ya que sirve para cambiar los permisos de un archivo ordinario y de un directorio. Existen dos formas de cambiar los permisos. Se pueden cambiar teniendo en cuenta los permisos existentes (modo simbólico), o se pueden asignar permisos independientemente de los ya existentes (modo absoluto).

Modo simbólico: cuando se utiliza el modo simbólico se pueden añadir o quitar permisos a los archivos y directorios. El formato del comando `chmod` simbólico es: `# chmod [who] código-operador permisos archivo`. Donde,

*who*, es el tipo de usuario y puede tener los siguientes valores:

- ➔ *u* : propietario del archivo.
- ➔ *g* : grupo del que el propietario es miembro
- ➔ *o* : usuarios clasificados como otros.
- ➔ *a* : todos los usuarios del sistema (propietario, grupo y otros)

*código-operador*, indica la operación que se va a realizar:

- ➔ *+* : añadir permisos.
- ➔ *-* : quitar permisos.

*permisos* ➔ *r* : permiso de lectura, *w* : permiso de escritura y *x* : permiso de ejecución.

*Archivo* ➔ nombre de archivo o directorio.

Por ejemplo, supongamos que el archivo *datos* tiene los siguientes permisos: `-rwxr--r--` y supongamos que queremos dar al grupo de usuarios y al resto de los usuarios del sistema, el permiso de ejecución; entonces se colocaría: `chmod go+x datos`

Modo absoluto: en este modo se especifica con 3 dígitos numéricos; cada número representa los permisos de cada tipo de usuario. Estos dígitos se obtienen, para cada clase de usuario, a partir de los valores siguientes:

4 : permiso de lectura.

2 : permiso de escritura

1 : permiso de ejecución.

Así tendremos:

0 : ningún permiso

1 : permiso de ejecución

2 : permiso de escritura

3 : permiso de ejecución y escritura (1+2)

4 : permiso de lectura

5 : permiso de lectura y ejecución (4+1)

6 : permiso de lectura y escritura (4+2)

7 : permiso de lectura, escritura y ejecución (4+2+1)

La sintaxis para el comando `chmod` absoluto es: *chmod modo archivo* donde:

- modo: son 3 dígitos numéricos. Cada uno de ellos corresponde a los permisos de cada tipo de usuario.
- Archivo, nombre de archivo o directorio.

Por ejemplo: `chmod 777 datos`, concede permisos de lectura, escritura y ejecución sobre el archivo `datos`, a todos los usuarios.

El **comando *chown*** se utiliza para cambiar el dueño y el grupo de un archivo. El dueño de un archivo solo lo puede cambiar el usuario `root` mientras que el grupo además de `root`, lo puede cambiar el propio dueño, siempre que pertenezca al nuevo grupo.

Sintaxis: *chown [opciones] dueño[:grupo] archivos*

*chown [opciones] :grupo archivos*

*Opción: -R en los directorios cambia el dueño y/o el grupo recursivamente.*

Ejemplos: *chown luis.grupopp tesis*, coloca a luis como propietario y como grupo "grupopp" del archivo tesis. *chown -R root /tmp/oculto* coloca a root como propietario del directorio "/tmp/oculto" y de todo su contenido.

Existe también el **comando *chgrp*** que se emplea de forma similar pero para cambiar exclusivamente el grupo. *chgrp ftp /usr/ftp*, coloca como grupo "ftp" del archivo "/usr/ftp"

## UNIDAD V - EDITOR VIM

### INTRODUCCIÓN A VIM

vi (visual editor) es un editor de texto realizado originalmente por William Joy para la versión de UNIX de la universidad de Berkeley: BSD. Posteriormente este potentísimo editor se incorporó al System V de AT&T convirtiéndose en herramienta estándar.

VIM (vi Improved) es un moderno editor de GNU compatible con vi pero con muchas funcionalidades añadidas, entre otras deshacer multinivel, multiventanas, multibufes, coloreado de sintaxis, autocompletado de nombres de archivos, ayuda en línea, apertura de directorios... Normalmente en los sistemas que disponen de Vim existe un enlace llamado vi para que esté disponible como tal.

El éxito de vi se debe a varios factores. El primero es que vi es un editor pensado para utilizar en pantalla completa; hay que tener en cuenta que antes los editores de texto solo mostraban la línea que estabas editando. De hecho vi está basado en el editor de líneas ex y se puede alternar entre un editor y otro mientras se trabaja o ejecutar comandos de ex en vi.

El segundo valor que popularizó vi es que todos sus comandos se realizan con el teclado alfanumérico permitiendo su uso indiscriminado entre terminales con distintas configuraciones. Así por ejemplo para mover el cursor no se utilizaban las flechas del teclado sino las letras h(izq)-j (abj)-k(arr)-l(der), aunque hoy día permite los dos sistemas. De nuevo hay que tener en cuenta que hasta el System V no existía apenas compatibilidad entre los distintos UNIX.

Hoy día los principales valores de vi son su universalidad (disponible en todos los sistemas UNIX/GNU), su eficiencia (es ligero y usa poca memoria) y sobre todo su potencia (no conozco nadie que sepa utilizar todas sus funciones, siempre se aprende alguna nueva). Esta potencia viene de la distinción entre modo comando y modo edición y en la versatilidad de los comandos; todos, por sencillos que parezcan, pueden combinarse con repeticiones, marcas, rangos de líneas y/o expresiones regulares e ir complicándose cuanto se desee, con gran versatilidad.

vi, como veremos, es un editor difícil de aprender, pero fácil de utilizar. A lo largo del texto empezaremos con versiones sencillas de los comandos e iremos complicando los ejemplos como

muestra de lo que se puede hacer. En las distribuciones GNU/Linux derivadas de Debian (Canaima, Ubuntu, entre otros.), para activar todas las funcionalidades de vim, hay que instalarlo mediante el siguiente comando: `# apt-get install vim`

## Generalidades

Para aprender a utilizar vim es bueno tener en cuenta dos cosas muy importantes. La fundamental, por hacerlo diferente a los demás editores, es que vim tiene dos modos básicos de funcionamiento, el modo comando y el modo edición (inserción o sustitución).

Cuando abrimos un archivo con vim (`vim nombre_archivo`) comenzamos en modo comando. Esto quiere decir que no podemos escribir. La primera sorpresa de un usuario novel con vim será que si empieza a escribir no aparecerá el texto. Es más, seguramente empiecen a ocurrir cosas 'raras' (se están ejecutando comandos).

También hay que tener en cuenta que los comandos distinguen entre mayúsculas y minúsculas. No es lo mismo el comando 'j' que el comando 'J'; no es lo mismo el comando 'p' que el comando 'P'; etc.

Para encontrar todas las funciones disponibles de vim puedes utilizar el comando 'man vim'.

## Iniciando vim

El modo de iniciar vim es: `vim [opciones] [nombres_archivo]`

Ejemplos:

- **vim prueba** → abrirá el archivo prueba o lo creará si no existe.
- **vim** → iniciará el editor vim creando un archivo nuevo aún sin nombre.
- **vim -R prueba** → abrirá el archivo prueba en modo 'solo lectura'.
- Otro modo de invocar vim es con **view** que equivale a **vim -R** e implica 'solo lectura'.
- En algunos sistemas también se puede llamar a **vedit** que equivale a **vim** con los modos 'novice', 'showmode' y 'nomagic' activos.
- Para el resto del manual supondré que el lector dispone mientras lo lee de un sistema con el editor vim para ir realizando los ejercicios que se comentan.



Ejecute *'vim prueba'*, el editor limpiará la pantalla. La primera línea aparece en blanco con el cursor. La última línea es la línea de estado en la que aparecerá "prueba" [New file]. Las demás líneas aparecen con el carácter '~' que indica que en esas líneas no hay ningún texto (líneas tras el final del archivo).

### ***Modo comando (introducción)***

Como ya se ha comentado al lanzar el editor comenzamos en 'modo comando'. Esto quiere decir que vim interpretará nuestras pulsaciones como mandatos, no como texto. Por ejemplo pulse las teclas Ctrl + G. El editor nos informará en la línea de estado sobre el archivo. En nuestro caso mostrará: *"prueba"*  
*line 1 of 1 -100%--*

Otra de las cosas que podemos hacer es utilizar los comandos del editor en línea ex, para ello utilizamos ':' (dos puntos). Por ejemplo, pulse ':' después 'q' (quit) e INTRO -en adelante ':q'- con lo que finalizaremos la sesión de trabajo del editor 'vim'.

Recuerde que en los comandos de ex (precedidos por ':') necesita pulsar INTRO ya que generalmente admiten parámetros, mientras que en los comandos 'visuales' se ejecutan inmediatamente (no necesita pulsar INTRO).

### ***Modo edición (introducción)***

Volvamos a vim (ejecute *'vim prueba'*). Pulsemos simplemente la tecla 'i' (significa insert). En apariencia no pasará nada ya que vim no indica el modo en que nos encontramos. Sin embargo en este momento vim interpretará nuestras pulsaciones como texto, no como mandatos, con las siguientes excepciones:

- **INTRO** -- Comienza una nueva línea.
- **BACKSPACE** -- Borra el carácter anterior.
- **CTRL-W** -- Borra la palabra anterior.
- **CTRL-U** -- Borra la línea actual.
- **CTRL-V** -- Permite la inserción de caracteres especiales (INTRO, ESC...)

Escriba:

*Sé fiel a tus verdaderas aspiraciones INTRO*

*pase lo que pase a tu alrededor. INTRO*

*Epicteto de Frigia. INTRO*

*Si buscas resultados distintos, INTRO*

*No hagas siempre lo mismo. INTRO*

*Albert Einstein*

Para finalizar el modo inserción pulse ESC (Tecla Escape).

## **La Tecla ESC**

El principal uso de la tecla ESC es finalizar el modo edición para volver al modo comando. Además se utiliza para cancelar un comando a mitad. En todo caso con dos pulsaciones de la tecla ESC nos encontraremos en modo comando y sin ningún comando a medias. Si ya se encuentra en modo comando sin ningún comando que cancelar y pulsa ESC, vim le avisará emitiendo un breve pitido.

Ante la duda pulse dos veces ESC y se encontrará en el modo comando. Una vez en modo comando pruebe a pulsar `':q'` vim le responderá con: ***No write since last change (:quit! Overrides)*** y no finalizará su sesión de trabajo.

vim le avisa que no ha guardado (escrito) el archivo tras los últimos cambios y también le indica como 'sobrepasar' los avisos.

! indica imperativo, y su significado lo veremos un poco más adelante.

Pulse `':'` después `'w'` (write) e INTRO -en adelante `':w'`-. vim responde:

"prueba" [New file] 5 lines, 178 characters

Ahora el archivo prueba se ha creado realmente en el sistema de archivos y contiene nuestra cita.

## ***Desplazamiento (introducción)***

Para desplazarnos por el texto debemos estar en modo comando. Podemos utilizar indistintamente las flechas de cursor o las letras h(izq)-j (abj)-k(arr)-l(der). También puede utilizar SPACE para avanzar y BACKSPACE para retroceder. Desplácese hasta el comienzo del texto.

## ***El imperativo !***

Ahora pulse 'i' y a continuación escriba: "Cita: INTRO" y pulse ESC. Pulse luego ':q' y como antes vim responde ***No write since last change (:quit! overrides)***

Ahora pulse ':q!'. Al hacerlo saldrá de vim sin guardar los últimos cambios. El imperativo también se usa para otros casos, como por ejemplo sobrescribir archivos existentes o salir sin haber editado todos los archivos abiertos (estos temas se verán más adelante).

## ***Configurando vim***

Se puede alterar el funcionamiento estándar de vim. Para ello pueden utilizarse archivos de configuración cuyo nombre y ubicación puede variar según el sistema. Los más habituales son ***.vimrc .viminfo .virc .exrc...***

Además puede utilizarse, por supuesto, el modo comando. Por ejemplo con ':showmode' vim nos indicará si estamos en modo comando (no indica nada) modo inserción (insert mode) o modo sustitución (replace mode). Estos dos últimos son variantes del modo edición. También se puede utilizar ':set number' para ver a la izquierda del texto el número de línea (útil para programar). Se quitan con ':set nonumber'. En caso de tener activado el coloreado de sintaxis puede ser útil ':set background=dark' o ':set background=light'

## ***Desplazamiento (movimientos más rápidos)***

Además de utilizar para mover el cursor las teclas vistas, podemos utilizar las siguientes:

- → inicio de la línea siguiente.
- 0 (cero) → inicio de la línea actual.
- \$ → final de la línea actual.
- - → Inicio de la línea anterior.
- ^F → página (pantalla) siguiente.
- ^B → página (pantalla) anterior.
- h → línea superior de la pantalla.

- m → línea del centro de la pantalla.
- l → línea inferior de la pantalla.
- b → comienzo de palabra anterior.
- w → comienzo de palabra siguiente.
- e → siguiente final de palabra.
- xG → línea x del texto, siendo x un número de línea.
- G → Última línea del texto

### *Algunas consideraciones sobre el texto*

vim distingue entre:

- Palabras → caracteres separados por espacios, tabuladores o saltos de línea.
- Expresiones → igual a las palabras pero considerando caracteres especiales (man vim para más información).
- Sentencias → caracteres entre los signos '.' (punto), '?' (interrogación) y '!' (exclamación) o saltos de línea.
- Líneas → texto entre dos saltos de línea (INTRO).
- Párrafos → texto separado por líneas en blanco.
- Además el final de archivo también termina todas ellas.

Principalmente los comandos de vim trabajan con palabras y líneas. Es importante tener en cuenta que una línea de texto puede ocupar más que el ancho de pantalla. En este caso vim nos la mostrará en varias líneas de pantalla, pero seguirá considerándose una única línea de texto.

Pulse ':set number' para que le muestre los números de línea.

Pulse 'lG' para situar el cursor al principio del texto. Pulse 'i' y escriba:

Esta es una línea larga de texto que ocupa más que el ancho de la pantalla, por lo que el editor me la mostrará en varias líneas de pantalla, siendo solo una línea de texto. INTRO ESC

El texto queda como sigue:

*1 Esta es una línea larga de texto que ocupa más que el ancho de la pantalla, por lo que el editor me la mostrará en varias líneas de pantalla, siendo solo una línea de texto.*

*2 Aquel que luche con monstruos*

*3 que procure no convertirse en un monstruo él mismo;*

*4 que tenga cuidado aquel que mire al abismo,*

*5 pues el abismo le devolverá la mirada.*

*6 F. Nietzsche*

Teclee '1G' para situarse al principio del texto y después pulse '^G' (Ctrl+g). El editor mostrará: "prueba" line 1 of 6 –16%--

También puede pulsar varias veces '+' y '-' para comprobar que el nuevo texto es una sola línea de texto, aunque se muestre en dos líneas de pantalla.

Pulse ':set nonumber' para que desaparezcan los números de línea.

### ***Borrando texto***

Estando en modo comando, el modo simple de borrar es utilizar:

- x → borra el carácter indicado por el cursor
- X → borra el carácter anterior al indicado por el cursor
- D → borra desde el carácter indicado por el cursor hasta el final de línea
- dd → borra la línea indicada por el cursor

El comando 'd' (delete) borra desde el carácter indicado por el cursor hasta lo indicado por el siguiente comando de desplazamiento. Así:

- dw -- borra hasta el comienzo de palabra siguiente
- db -- borra hasta el comienzo de palabra anterior

- dxG -- borra hasta la línea x del texto, siendo x un número de línea
- dG -- borra hasta la última línea del texto
- d\$ -- borra hasta el final de línea (equivale a 'D')

Estando al principio del texto pruebe a borrar algunos caracteres y un par de palabras. Por último borre toda la línea con 'dd'.

### ***Modo edición (un poco más)***

Hasta ahora solo se ha visto el modo más sencillo de insertar texto con el comando 'i'. Este comando permite insertar delante del texto indicado por el cursor. Sin embargo existen multitud de comandos para cambiar al modo edición, distinguiendo entre modo inserción (el texto que escribimos se añade al existente) y modo reemplazo (el texto que escribimos sustituye a texto existente). Los comandos principales son:

- a (append) → permite insertar texto detrás del texto indicado por el cursor.
- I (Insert) → permite insertar texto al inicio de la línea indicada por el cursor.
- A (Append) → permite insertar texto al final de la línea indicada por el cursor.
- o → añade una línea después de la actual y permite insertar texto.
- O → añade una línea antes de la actual y permite insertar texto.
- r (replace) → sustituye el carácter indicado por el cursor por otro carácter. (No hace falta pulsar ESC para volver al modo comando).
- R (Replace) → permite sobrescribir el texto a partir del carácter indicado por el cursor.
- s (substitute) → sustituye el carácter indicado por el curso por el texto introducido.
- S (Substitute) → sustituye la línea indicada por el cursor por el texto introducido.
- C (Change) -- sustituye desde el carácter indicado por el cursor hasta el final de línea.

El comando 'c' (change) cambia el texto desde el carácter indicado por el cursor hasta lo indicado por el siguiente comando de desplazamiento. Así:

- cw → cambia hasta el comienzo de palabra siguiente
- cb → cambia hasta el comienzo de palabra anterior
- cxG → cambia hasta la línea x del texto, siendo x un número de línea
- cG → cambia hasta la última línea del texto
- c\$ → cambia hasta el final de línea (equivale a 'C')

Teclee '1G' para situarse al principio del texto y después pulse 'o' y escriba 'puede salir escaldado' y pulse ESC. Después pulse 'O' y escriba 'cuidado porque' y pulse ESC. Ahora pulse 'I' y escriba 'tenga mucho' y ESC. El texto habrá quedado como sigue:

*Aquel que luce con monstruos*

*tenga mucho cuidado porque*

*puede salir escaldado*

*que procure no convertirse en un monstruo él mismo;*

*que tenga cuidado aquel que mire al abismo,*

*pues el abismo le devolverá la mirada.*

*F. Nietzsche*

### **Otros comandos útiles**

- Uno de los comandos más útiles en cualquier editor es el comando 'deshacer' (undo). En vim el comando 'u' deshace el último cambio. Si se vuelve a pulsar 'u' deshace la acción de deshacer, es decir pulsar 'uu' hace que el texto quede igual.
- Pulse 'u' y desaparecerá 'tenga mucho' en la segunda línea. Vuelva a pulsar 'u' varias veces para comprobar su efecto.
- Vim tiene 'deshacer multinivel' lo que permite, según la configuración, que al pulsar varias veces 'u' se deshagan los últimos cambios que se han ido realizando.
- El comando 'J' elimina el salto de línea de la línea actual. De este modo une la línea siguiente al

final de la línea actual.

- El comando `':w'` escribe (guarda) el texto actual. También se puede utilizar `':w nombre_archivo2'` con lo que guardará el texto en el archivo `nombre_archivo2` (aunque seguiremos trabajando con `nombre_archivo`). `':2,5w nombre_archivo2'` guarda las líneas 2 a 5 en `nombre_archivo2`.
- También se puede combinar el comando `':w'` con el comando `':q'` escribiendo `':wq'`. El comando `':x'` equivale a `':wq'` (guardar los cambios y salir).
- El comando `':r nombre_archivo3'` lee el archivo `nombre_archivo3` y lo escribe después de la línea indicada por el cursor. `':5r nombre_archivo3'` lo inserta en la línea 5.
- `^g` (Ctrl+g) -- muestra información sobre el archivo, así como el número de línea (muy útil para desplazarse a una línea copiar desde donde estamos hasta la línea x, etc.)
- `^l` (Control+L (ele)) -- redibuja la pantalla. Muy útil cuando el terminal hace que el scroll del texto monte las líneas y ya no sabes si lo que ves es realmente lo que pone

### ***Repeticiones de comandos***

Una de las posibilidades más versátiles de vim es la repetición de cualquiera de sus comandos. Hay dos modos básicos de hacerlo.

- A posteriori: `.` (punto) → repite el último comando de modificación del texto. Este último comando puede haber sido cambiar una palabra por una frase, añadir un texto a final de línea, borrar un párrafo, insertar una tabulación.
- A priori, se puede pulsar un número antes de ejecutar un comando y éste se repetirá tantas veces como hayamos indicado.

Ejemplos:

`5+`, nos llevará al inicio de la 5a línea después de la indicada por el cursor.

`3Dd`, borrará 3 líneas.

`3J`, unirá tres líneas.

`8llaESC`, insertará 8 veces la palabra *la* (insertará 'lalalalalalalala').



2s(texto)ESC, sustituye dos caracteres por el texto introducido.

2cw(texto)ESC, sustituye dos palabras por el texto introducido.

## UNIDAD VI - INSTALACIÓN DE PAQUETES

### ADVANCED PACKAGING TOOL (APT)

apt es un conjunto de herramientas avanzadas de gestión de paquetes; hace muchas más que synaptic ya que esta última es una herramienta gráfica basada en un subconjunto de comandos de apt. Los nombres, versiones y descripciones de los paquetes disponibles se guardan en una base de datos en el directorio */var/cache/apt*.

Antes de usar apt y los programas que dependen de él, hay que indicarle dónde debe buscar los paquetes Debian. Las fuentes de paquetes se indican en el archivo */etc/apt/sources.list*; estas fuentes se denominan *repositorios*.

Si escribimos en un terminal el comando:

```
# cat /etc/apt/sources.list

##

deb cdrom:[Debian GNU/Linux 4.0 r0 _Etch_ - Official i386 NETINST Binary-1
20070407-11:29]/ etch contrib main

#deb cdrom:[Debian GNU/Linux 4.0 r0 _Etch_ - Official i386 NETINST Binary-1
20070407-11:29]/ etch contrib main

#deb http://192.168.1.100/debian/mirror/ stable main contrib non-free

#deb http://ftp.es.debian.org/debian stable main contrib non-free

deb file:/mnt/debian/mirror stable main contrib non-free

#deb http://http.us.debian.org/debian stable main contrib non-free

# Line commented out by installer because it failed to verify:

#deb http://security.debian.org/ etch/updates main contrib
```

*# Line commented out by installer because it failed to verify:*

*#deb-src <http://security.debian.org/> etch/updates main contrib*

Las líneas que comienzan con “#” son ignoradas. Todas empiezan por “deb”, seguido de la URL del repositorio, y a continuación los directorios en los que se buscarán los paquetes. Alternativamente se puede añadir un CD con paquetes, para cargarlos se utiliza el comando: `# apt-cdrom add` automáticamente se añaden a la base de datos los paquetes del CD.

El sistema apt incluye herramientas que permiten actualizar un gran número de paquetes simultáneamente, incluso renovar la distribución completamente. Para actualizar la lista de paquetes de todos los *repositorios* se puede utilizar la orden: `# apt-get update` y para actualizar a las nuevas versiones todos los paquetes de los que exista una versión nueva: `# apt-get upgrade`.

Si la actualización supone un cambio a una versión completamente nueva de la distribución necesitamos: `# apt-get dist-upgrade`. Puede tardar mucho tiempo en instalar las listas de paquetes en estas dos últimas operaciones. `apt-get install`.

La instalación y desinstalación de paquetes con apt es sencilla. Basta con ejecutar el comando `apt-get` con el argumento “install” y la lista de paquetes a instalar. `# apt-get install "paquetes"`. apt instalará los paquetes seleccionados y cualquier otro del que estos dependan.

Para la instalación considerará todas las fuentes del archivo *sources.list*, obteniendo las versiones más actualizadas de los paquetes. Si un paquete se encuentra en varias fuentes, utilizará aquella que aparezca en primer lugar en el archivo. Si estamos interesados en descargar un paquete pero no queremos instalarlo, tenemos que incluir la opción “-d”, de esta forma: `# apt-get -d install "paquete"`

Para desinstalar paquetes se emplea la opción “remove” de apt-get. Por defecto se conservan los archivos de configuración del paquete que se desinstala, pero se puede indicar que se borren con la opción “--purge”. Para desinstalar el paquete fortune-es sin eliminar los archivos de instalación: `# apt-get remove fortune-es`. Y si quisiéramos eliminar también los archivos de instalación: `# apt-get --purge remove fortune-es`.

Los paquetes descargados se guardan por defecto en `/var/cache/apt/archives`, por si se necesita utilizarlos en otra ocasión. Es aconsejable eliminar los paquetes que ya se han instalado y no son necesarios mediante el comando: `# apt-get clean`

Existe un programa disponible referente a apt, llamado apt-cache, que permite obtener información diversa sobre los paquetes y permite realizar búsquedas, tanto en los nombres como en las descripciones. Para ver la información referente a un paquete (nombre, versión, dependencias, descripción, ...) se utiliza el parámetro “show”: # apt-cache show dselect

Si por el contrario queremos buscar todos los paquetes relacionados con “dselect” usaríamos: #apt-cache search dselect y se obtendría:

apt - Advanced front-end for dpkg

apt-zip - Update a non-networked computer using apt and removable media

aptitude - terminal-based apt frontend

checksecurity - basic system security checks

cron - management of regular background processing

debarchiver - Tool to handle debian package archives

dpkg - Package maintenance system for Debian

dpkg-ftp - Ftp method for dselect.

dpkg-multicd - Installation methods for multiple binary CDs

dselect - a user tool to manage Debian packages

grep-dctrl - Grep Debian package information

isdnutils - Most important ISDN-related packages and utilities que son los paquetes disponibles que contienen, en el nombre o en su descripción, la palabra: “dselect”.

## **APTITUDE**

Aptitude es un envoltorio que trabaja sobre apt. No es gráfico, sino que tiene una interfaz con debconf y también puede usarse en línea de comandos. ¿Qué lo diferencia de apt?

- Automatización de dependencias, en el sentido de que, si un paquete depende de la librería libX, y en una versión futura deja de hacerlo, aptitude eliminará libX en ese momento (y comprobará que nadie más la está usando).
- Posibilidad de ver mucha información de un paquete de manera sencilla. Por ejemplo, ¿tienes la posibilidad de actualizar un paquete pero no sabes si te interesa? Con aptitude puedes ver el

changelog de la nueva versión disponible antes de instalarla. ¿Quieres ver qué paquetes dependen de uno dado?

- Resolución inteligente de conflictos.

Sintaxis: *#aptitude install nombre\_paquete*. Por ejemplo, *#aptitude install nmap*

## SYNAPTIC

Para la instalación del programa Synaptic sigue los pasos del Programa Synaptic.

## DPKG

Es una herramienta que permite la manipulación directa de archivos “.deb”. Gracias a la existencia de apt, sólo es necesario recurrir a ella cuando nos encontramos con paquetes sueltos, es decir, que no forman parte de ningún repositorio. También es útil si hemos usado aptget con la opción “-d” y llevamos los paquetes descargados a otro PC para instalarlos. Para instalar un paquete llamado “paquete.deb” la forma de hacerlo es:

```
# dpkg --install paquete.deb
```

En el caso de que se produzcan problemas de dependencias el paquete no será configurado pero sí desempquetado; es necesario instalar los paquetes para resolver estas dependencias de otros paquetes. Si ese es el caso, una vez instalados todos hay que ejecutar:

```
# dpkg --configure --pending
```

De esta forma se configuran los paquetes pendientes de configuración. Hay que ser muy cuidadoso con los paquetes instalados directamente con dpkg, ya que la lista de dependencias puede ser tan grande que haga inviable la instalación a mano (por eso se creó apt).

Para desinstalar paquetes se emplea: *# dpkg --remove paquete*. Y si también se quieren eliminar los archivos de configuración: *# dpkg --purge paquete*. Nótese que en ambos casos hay que poner el nombre del paquete, no el nombre del archivo .deb.

## **DSELECT**

Es una interfaz para manejar *dpkg*. Para verla abre un terminal y ejecuta “dselect”.

## **ALIEN**

Es un paquete que convierte entre Red Hat rpm, Debian .deb, Slackware .slp y pkg en Solaris. En formatos de archivos, si por alguna razón quieres usar un paquete de otro distribuidor en linux, que no sea de tu sistema actual, puedes usar el comando alien para convertirlo al formato que corresponde a tu sistema. Pero alien no debe ser usado para suplantar paquetes importantes como init, libc y otras cosas que no son necesarias para tu sistema. Ejemplo para convertir paquete de RedHat rpm a debian: *#alien nombre\_paquete.rpm*, nos generará un *nombre\_paquete.deb*

Es muy típico aplicar el comando alien para convertir controladores de impresoras preparados en \*.rpm a paquetes \*.deb

## UNIDAD VII - KERNEL

### DEFINICIÓN DE KERNEL

El Kernel es el núcleo del sistema operativo. El actúa como un intérprete entre el usuario y el hardware. El kernel controla el acceso a los recursos de hardware de la computadora y determina como compartir estos recursos de una manera equitativa. El incluye los drivers del hardware, sistema de archivos, redes, manejo de memoria y administración de los procesos.

Virtualmente, el Kernel puede ser configurado y optimizado para cualquier entorno, a través de recopilación del kernel mismo. Podrías querer recompilar el kernel para incluir drivers para hardware específico o para actualizar drivers para la corrección de errores o para incluir nuevas características.

El Kernel es de los primeros software a ejecutarse en un computador. Una vez el kernel ha terminado su iniciación, hace un llamado al proceso init (llamado el proceso padre de todos los procesos). El kernel provee todas las funcionalidades básicas a los programas así como el manejo de los recursos del sistema: hardware, procesos, memoria, I/O y sistema de archivos. La funcionalidad del kernel es mejorada sumándole/removiéndole código compilado llamado módulos o manejadores (drivers) de dispositivos.

#### ***Tipo de kernel***

El kernel de Linux es un proyecto activo con un desarrollo continuado. En este proceso existen dos ramificaciones que viajan en paralelo. La primera es la denominada versión estable del kernel y su intención es para producción solamente y no investigación. La otra es la versión de desarrollo y es donde los desarrolladores denominados hackers prueban y analizan propuestas de mejoras. Casi siempre es inestable, con problemas y características incompletas. Puedes reconocer el tipo de kernel por sus números de versión. El formato de este número es: X.Y.Z, donde X es la versión mayor y es la menor y Z es el nivel de patch o revisión de mejora. Si la versión menor es impar entonces es desarrollo (developers) y si es par entonces es estable.

#### ***Obteniendo un nuevo kernel***

Antes de poder compilar el kernel, primero debes desempacar el código fuente en el sitio adecuado y

preparar el directorio fuente. Para obtener los fuentes de un kernel hay diferentes mecanismos o lugares para encontrarlos. Una opción es vía ftp anónimo en ftp.kernel.org. En Debian 2.2 otra opción es mediante los paquetes de la distribución. Lo más habitual es que vengan empaquetados en un único fichero de nombre linux-x.y.z.tar.gz o bien con sufijo bz2 lo cual indica que han sido comprimidos con bzip2.

### ***Preparación de un nuevo kernel***

Una vez obtenidos los fuentes hay que desempaquetarlos. Para ello hay que colocar el fichero comprimido con los fuentes en el directorio /usr/src y, aquí, proceder a descomprimirlos bien con gunzip (si tiene sufijo gz) o con bunzip2 (si tienen sufijo bz2).

Ejemplo: host:/usr/src# bunzip2 kernel-source-2.4.5.tar.bz2, una vez descomprimidos hay que deshacer el tar, host:/usr/src# tar xvf kernel-source-2.4.5.tar

Ejemplo: host:/usr/src# bunzip2 x.y.z.tar.bz2, una vez descomprimidos hay que deshacer el tar, host:/usr/src# tar xvf kernel-source-x.y.z.tar

### ***Configurando el kernel***

Este es el paso más delicado en todo el proceso de compilación de un nuevo kernel ya que podría ocurrir que incluso no pudiera arrancarse el sistema si no se selecciona alguna opción fundamental para el mismo. Una vez que el superusuario se sitúa en el directorio /usr/src/linux/ donde se encuentra toda la estructura de directorios y ficheros que contienen los fuentes del kernel, se puede ejecutar el comando: host:/usr/src/linux# make config

Sin embargo, existen otros modos más adecuados y fáciles de usar, en concreto en modo texto el más ampliamente usado es: host:/usr/src/linux# make menuconfig. Si está operativo el sistema gráfico X-window, es más cómodo usar: host:/usr/src/linux# make xconfig

Dentro de estos scripts hay que responder con si (y), con no (n) o con la opción m de módulo. La opción si significa que este módulo se compilará dentro del kernel, la opción no indica que no se compilará, mientras que la opción m significa que se compilará el módulo pero que no se incluirá dentro del kernel.



## ***Objetivos de Compilación***

La mayoría de las veces no será necesario puesto que el núcleo estándar entregado con Debian gestiona la mayoría de configuraciones. Además, Debian ofrece habitualmente varios núcleos alternativos. Así, debería comprobar si hay un paquete de imagen de núcleo alternativa que se ajuste mejor a su hardware. En cualquier caso es útil compilar un nuevo núcleo para:

- Tratar necesidades especiales de hardware, o conflictos de los mismos con núcleos predeterminados.
- Utilizar opciones del núcleo que no están soportadas en los núcleos preparados, como puede ser el caso del soporte de memoria elevada (más de 4GB).
- Otimizar el núcleo eliminando controladores no usados para acelerar el tiempo de arranque.
- Crear un núcleo monolítico en lugar de uno modular.
- Ejecutar un núcleo actualizado o de desarrollo.
- Aprender más de los núcleos de Linux.

## ***Compilar un nuevo núcleo***

Una vez finalizada la fase de configuración, cuando grabamos o almacenamos esta configuración el propio script nos pide que ejecutemos los siguientes comandos consecutivamente: *host:/usr/src/linux # make dep ; make clean*

*make dep* asegura que todas las dependencias, como por ejemplo los include se encuentran en su sitio. *make clean* borra todos los ficheros objeto que hubieran sido generados en una fase de compilación anterior. Tras las dependencias y la limpieza hay que compilar el kernel.

## ***Gestión de la imagen del núcleo***

Para ello se ejecuta el comando: *host:/usr/src/linux# make bzImage* o bien: *host:/usr/src/linux# make bzdisk*

El primero compilará el kernel y lo guardará en fichero llamado *bzimage* en el directorio */usr/src/linux/arch/i386/boot/*. El segundo hace lo mismo pero, además, guarda el kernel en un dispositivo de almacenamiento. Esta opción permite que se pueda arrancar el sistema con el nuevo kernel sin tener

que instalar el kernel en el sector de arranque del disco duro, de tal forma que si existiesen problemas de funcionamiento pudiera volver a arrancarse el sistema con el kernel anterior.

### ***Instalando el nuevo kernel***

Para poder arrancar con el nuevo kernel es necesario instalarlo en el sector de arranque adecuado con un programa gestor de arranque. El más conocido es lilo. En este proceso se tiene que copiar el kernel al fichero `/vmlinuz` o bien hacer un link simbólico. Además, conviene pasar el antiguo kernel a `/vmlinuz.old` para dejar activos tanto el antiguo como el nuevo, para poder arrancar desde cualquiera de los dos. Una vez copiados o convenientemente enlazados se debe actualizar el fichero `/etc/lilo.conf`:

```
...  
image=/vmlinuz  
label=Linux  
read-only  
# restricted  
alias=deb  
image=/vmlinuz.old  
label=LinuxOLD  
read-only  
optional  
# restricted  
alias=2  
...
```

Una vez modificado este fichero de configuración del gestor de arranque lilo, se debe ejecutar el comando lilo para que se instale la nueva configuración en el sector de arranque: `# lilo`

## UNIDAD VIII - MANEJO Y TIPOS DEL SISTEMA DE ARCHIVOS

### CONSIDERACIONES AL MOMENTO DE HACER UN FILE SYSTEM

Hay que tener en cuenta que en GNU/Linux no existen las unidades de discos ni dispositivos de almacenamiento (floppys, USBSticks, pendrives, etc.), propiamente dicho por esta razón, antes de que se pueda utilizar un sistema de archivos, debe ser montado. El sistema operativo realiza entonces operaciones de mantenimiento para asegurarse que todo funciona. Como todos los archivos en UNIX están en un mismo árbol de directorios, la operación de montaje provocará que el contenido del nuevo sistema de archivos aparezca como el contenido de un subdirectorio existente en algún sistema de archivos ya montado.

### DISPOSITIVOS EN LINUX

UNIX, y por lo tanto GNU/Linux, reconocen dos tipos de dispositivos: dispositivos de bloques de accesoaleatorio (tales como discos), y dispositivos de caracteres (tales como cintas y líneas seriales), algunos de estos últimos pueden ser de acceso secuencial y algunos de accesos-aleatorio. Cada dispositivo soportado en GNU/Linux es representado en el sistema de archivos como un archivo de dispositivo.

Cuando lea o escriba sobre un archivo de dispositivo, los datos van o vienen desde el dispositivo que este representa. De esta manera no se necesitan programas especiales (y no se necesitan ningún método especial de programación, como descubrir interrupciones o escudriñar puertos seriales) para acceder a los dispositivos. Por ejemplo, para enviar un archivo a la impresora, puede simplemente ejecutar: *\$ cat filename > /dev/lp1* y el contenido del archivo es impreso (en este caso, el archivo debe estar en un formato que la impresora comprenda). Note, que no es una buena idea tener a varias personas realizando cat de sus archivos a la impresora al mismo tiempo. Generalmente se utiliza un programa especial para enviar los archivos a que sean impresos (usualmente lpr). Estos programas se aseguran de que solo un archivo esté siendo impreso en un momento dado, y automáticamente envía archivos a la impresora en cuanto se finalice la impresión del archivo previo. Algo similar puede ser necesario para la mayoría de los dispositivos. De hecho, raramente los archivos de dispositivos son utilizados

directamente.

Desde que los archivos de dispositivos se muestran como archivos en el sistema (en el directorio /dev), es fácil ver cuales de ellos existen, utilizando ls o algún otro comando similar. En la salida del comando ls -l, la primera columna indica el tipo de archivo y sus permisos. Por ejemplo, observe la salida al inspeccionar un archivo de dispositivo de un puerto serial:

```
$ ls -l /dev/ttyS0
```

```
crw-rw-r-- 1 root dialout 4, 64 Aug 19 18:56 /dev/ttyS0
```

El primer caracter en la primera columna arriba, es decir 'c' en crw-rw-r--, le indica el tipo de archivo, en este caso un dispositivo de caracteres. Para archivos comunes el primer carácter es '-', para directorios es 'd' y para dispositivos de bloques es 'b'. Examine la página de manual de ls para obtener información mas detallada. Note que usualmente todos los archivos de dispositivos existen, aún cuando el dispositivo por si mismo no se encuentre instalado. Esto quiere decir que aunque exista un archivo llamado /dev/sda, no significa que realmente haya un disco SCSI instalado. Tener todos los archivos de dispositivos facilita la instalación de los programas, y la incorporación de nuevo hardware (no existe *Dispositivos IDE en Linux* la necesidad de tener que conocer los parámetros correctos para crear los archivos de dispositivos de los nuevos dispositivos).

### ***Dispositivos IDE en Linux***

Cada disco rígido es representado por un archivo de dispositivo separado. Los archivos de dispositivos para los discos rígidos IDE son /dev/hda, /dev/hdb, /dev/hdc, y dev/hdd, respectivamente

### ***Dispositivos SCSI en Linux***

Los archivos de dispositivos para los discos rígidos SCSI son /dev/sda, /dev/sdb, etc.

## **PARTICIONES**

Un disco duro puede dividirse en varias *particiones*. Cada partición funciona como si fuera un disco duro independiente. La idea es que si sólo se tiene un disco, y se quieren tener, digamos, dos sistemas operativos en él, se pueda dividir el disco en dos particiones. Cada sistema operativo utilizará su propia partición tal y como se desea, y no tocará la otra. De esta forma los dos sistemas operativos pueden coexistir pacíficamente en el mismo disco duro. Sin particiones se tendría que comprar un disco duro

para cada sistema operativo.

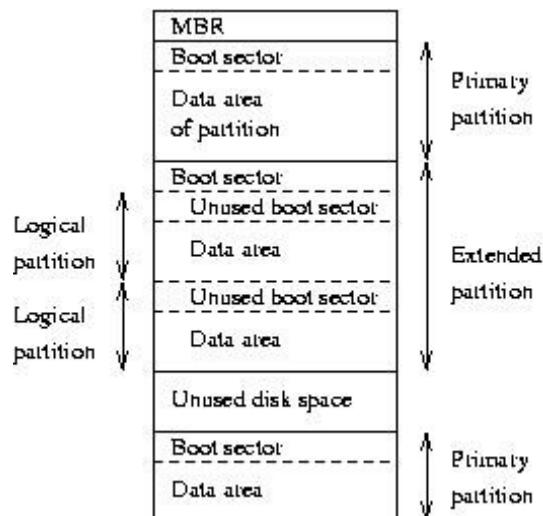
Los disquetes generalmente no se particionan. No hay ninguna razón técnica para ello, pero dado que son tan pequeños, particionarlos sería útil sólo en extrañas ocasiones. Los CD-ROM tampoco se suelen particionar, ya que es más fácil utilizarlos como un disco grande, y raramente existe la necesidad de tener varios sistemas operativos en uno de ellos.

### ***Particiones primarias y lógicas***

El esquema original de particionamiento para discos duros de PC permitía solamente cuatro particiones. Esto rápidamente se volvió demasiado escaso para la vida real, en parte porque algunas personas querían más de cuatro sistemas operativos (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, o Windows/NT,mpor nombrar algunos), pero principalmente porque algunas veces es buena idea tener varias particiones para un sistema operativo. Por ejemplo, el espacio swap está generalmente mejor colocado para Linux en su propia partición en lugar de la partición principal por cuestiones de rapidez.

Para superar este problema de diseño, se inventaron las *particiones extendidas*. Este truco permite particionar una *partición primaria* en sub-particiones. Esta partición primaria subdividida es la *partición extendida*; las sub-particiones son las *particiones lógicas*. Se comportan como particiones primarias, pero se crean de diferente manera. No existen diferencias de rendimiento entre ellas.

La estructura de particiones de un disco duro debe parecerse a la que aparece en la siguiente Figura, “*A sample hard disk partitioning*.”. El disco se divide en tres particiones primarias, la segunda de las cuales se divide en dos particiones lógicas. Una parte del disco no está particionada. El disco como un todo y cada partición primaria tienen un sector de arranque.



**Figura: Esquema simple de Particionado**

### ***Crear Particiones***

Existen muchos programas para crear y eliminar particiones. La mayoría de sistemas operativos tienen el suyo propio, y es buena idea utilizar el propio con cada sistema operativo, por si se diera el caso que haga algo inusual que los otros no puedan hacer. Muchos de estos programas se llaman fdisk, incluido el de Linux, o variaciones de esa palabra. Los detalles de uso del fdisk de Linux se dan en su página de manual. El comando cfdisk es similar a fdisk, pero tiene una interfaz más agradable (a pantalla completa).

Cuando se utilizan discos EIDE, la partición de arranque (la partición con los archivos de imagen del núcleo arrancable) debe estar completamente definida en los primeros 1024 cilindros. Esto es así porque el disco se utiliza a través de la BIOS durante el arranque (antes de que el sistema entre en el modo protegido), y la BIOS no puede manejar más de 1024 cilindros.

A veces es posible utilizar una partición de arranque que parcialmente se encuentre dentro de los primeros 1024 cilindros. Esto funciona siempre que todos los archivos que lee la BIOS se encuentran en los primeros 1024 cilindros.

Como esto es difícil de conseguir, es *mala idea* intentarlo; nunca se sabe si una actualización del núcleo o una desfragmentación del disco originará un sistema no arrancable. Así que asegúrese que su

partición de arranque está completamente colocada dentro de los primeros 1024 cilindros. 8 .

Algunas versiones modernas de BIOS y discos IDE pueden, de hecho, manejar discos de más de 1024 cilindros, Si tiene uno de esos sistemas, se puede olvidar del problema; si no está seguro, coloque la partición de arranque en los primeros 1024 cilindros.

Cada partición debe tener un número par de sectores, puesto que el sistema de archivos Linux utiliza un tamaño de bloque de 1 kilobyte , es decir, dos sectores. Un número impar de sectores provocará que el último no pueda utilizarse. Esto no es ningún problema, pero resulta feo, y algunas versiones de fdisk avisarán de ello.

Cambiar el tamaño de una partición requiere que primero se realice una copia de seguridad de todo lo que quiera salvar de esa partición (a ser posible de todo el disco, por si acaso), borrar la partición, crear una nueva, y entonces restaurarlo todo a la nueva partición. Si la partición crece, puede necesitar ajustar los tamaños (y guardar y restaurar) las particiones adjuntadas también.

Como cambiar una partición es doloroso, es preferible establecerlas correctamente al principio, o tener un rápido y fácil de utilizar sistema de copia de seguridad. Si está instalando desde un medio que no requiere demasiada intervención humana (digamos, desde un CD-ROM, en contraposición a los disquetes), es generalmente más fácil probar con diferentes configuraciones al principio. Como no tiene todavía datos que guardar, no es tan costoso el modificar particiones varias veces.

Existe un programa de MS-DOS, llamado fips9 que redimensiona una partición MS-DOS sin tener que guardar y restaurar, pero para otros sistemas de archivos es todavía necesario.

Es importante resaltar que existe un programa muy útil para Linux llamado ntfs-resize que pertenece al proyecto Linux-ntfs. Puede redimensionar de manera segura y no destructiva particiones de tipo NTFS. Además soporta todas las versiones de NTFS y trabaja aún cuando el sistema de archivos se encuentre fragmentado.

## **SISTEMA DE ARCHIVOS**

Un sistema de archivos se crea, esto es, se inicia, con el comando mkfs. Existen en realidad programas separados para cada tipo de sistemas de archivos. mkfs es únicamente una careta que ejecuta el programa apropiado dependiendo del tipo de sistemas de archivos deseado. El tipo se selecciona con la

opción -t fstype.

Los programas a los que -t fstype llama tienen líneas de comando ligeramente diferentes. Las opciones más comunes e importantes se resumen más abajo; vea las páginas de manual para más información.

- -t fstype, selecciona el tipo de sistema de archivos.
- -c, busca bloques defectuosos e inicia la lista de bloques defectuosos en consonancia.
- -l filename, lee la lista inicial de bloques defectuosos del archivo dado.

## ***Herramientas del Sistema de Archivos***

### *Comprobar la integridad de un sistema de archivos con fsck*

Los sistemas de archivos tienden a ser propensos a los errores. La corrección y validación de un sistema de archivos puede ser comprobada utilizando el comando *fsck*. Puede ser instruido para reparar cualquier problema menor que encuentre, y alertar al usuario si hay errores irreparables. Afortunadamente, el código implementado en los sistemas de archivos puede estudiarse de forma muy efectiva, así que escasamente hay problemas, y normalmente son causados por fallos de alimentación, hardware defectuoso, o errores de operación; por ejemplo, no apagar el sistema adecuadamente.

La mayoría de los sistemas se configuran para ejecutar *fsck* automáticamente durante el arranque, así que cualquier error se detecta (y esperemos que corregido) antes que el sistema se utilice.

Utilizar un sistema de archivos corrupto tiende a empeorar las cosas: si las estructuras de datos se mezclan, utilizar el sistema de archivos probablemente las mezclará aún más, resultando en una mayor pérdida de datos.

En cualquier caso, *fsck* puede tardar un tiempo en ejecutarse en sistemas de archivos grandes, y puesto que los errores casi nunca suceden si el sistema se ha apagado adecuadamente, pueden utilizarse un par de trucos para evitar realizar comprobaciones en esos casos. El primero es que si existe el archivo */etc/fastboot*, no se realizan comprobaciones. El segundo es que el sistema de archivos ext2 tiene una marca especial en su superbloque que indica si el sistema de archivos se desmontó adecuadamente después del montaje previo. Esto permite a *e2fsck* (la versión de *fsck* para el sistema de archivos ext2) evitar la comprobación del sistema de archivos si la bandera indica que se realizó el desmontaje (la suposición es que un desmontaje adecuado indica que no hay problemas). Que el truco de */etc/ fastboot*



funcione en su sistema depende de sus guiones (scripts) de inicio, pero el truco de ext2 funciona cada vez que utilice e2fsck. Debe ser sobrepasado explícitamente con una opción de e2fsck para ser evitado.

La comprobación automática sólo funciona para los sistemas de archivos que se montan automáticamente en el arranque. Utilice fsck de forma manual para comprobar otros sistemas de archivos, por ejemplo, disquetes. Si fsck encuentra problemas irreparables, necesita conocimientos profundos de cómo funciona en general un sistema de archivos, y en particular el tipo del sistema de archivos corrupto, o buenas copias de seguridad. Lo último es fácil (aunque algunas veces tedioso) de arreglar, el precedente puede solucionarse a través de un amigo, los grupos de noticias y listas de correo de Linux, o alguna otra fuente de soporte, si no sabe cómo hacerlo usted mismo. Me gustaría contarle más sobre el tema, pero mi falta de formación y experiencia en este asunto me lo impiden. El programa de Theodore Ts'o debugfs puede ser de ayuda.

fsck debe ser utilizado únicamente en sistemas *dLuchar contra la Fragmentación* archivos desmontados, nunca en sistemas de archivos montados (a excepción del raíz en sólo-lectura en el arranque). Esto es así porque accede al disco directamente, y puede por lo tanto modificar el sistema de archivos sin que el sistema operativo se percate de ello. Habrá problemas, si el sistema operativo se confunde.

### *Comprobar errores en el disco mediante badblocks*

Puede ser buena idea comprobar los bloques defectuosos periódicamente. Esto se realiza con el comando badblocks. Saca una lista de los números de todos los bloques malos que puede encontrar.

Esta lista puede introducirse en *fsck* para grabar en el sistema de archivos las estructuras de datos para que el sistema operativo no intente utilizar los bloques malos para almacenar datos.

### *Luchar contra la Fragmentación*

Cuando un archivo se escribe en el disco, no puede escribirse siempre en bloques consecutivos. Un archivo que no está almacenado en bloques consecutivos está *fragmentado*. Leer un archivo fragmentado requiere mayor tiempo, puesto que la cabeza de lectura-escritura del disco debe moverse más. Es deseable evitar la fragmentación, aunque es un problema menor en un sistema con un buen caché buffer con lectura progresiva.

El sistema de archivos ext2 intenta mantener la fragmentación al Comprobar la integridad de un

sistema de archivos con fsck mínimo, manteniendo todos los bloques de un archivo juntos, incluso cuando no pueden almacenarse en sectores consecutivos. Ext2 efectivamente localiza el bloque libre más cercano a los otros bloques del archivo. Por lo tanto para ext2 hay poca necesidad de preocuparse por la fragmentación. Existe un programa para desfragmentar un sistema de archivos ext2, llamado extrañamente defrag.

Existen muchos programas de desfragmentación MS-DOS que mueven los bloques por todo el sistema de archivos para eliminar la fragmentación. Para otros sistemas de archivos, la desfragmentación debe hacerse guardando el sistema de archivos, volverlo a crear, y restaurando los archivos de la copia guardada. Guardar un sistema de archivos antes de desfragmentarlo es una buena idea para cualquier sistema de archivos, puesto que muchas cosas pueden ir mal durante la desfragmentación.

### ***Montar un Sistema de Archivos***

Ya se ha visto que Linux accede a los dispositivos mediante archivos (directorios de /dev), y, por este motivo, en Linux no hay el concepto de unidades, ya que todo está bajo el directorio principal . En Linux no se accede a la primera disquetera mediante la orden A: como en DOS sino que hay que *montarla*.

De este modo, tenemos dos conceptos nuevos:

- *montar*: decirle a Linux que se va a utilizar un determinado dispositivo con un determinado sistema de archivos y estará en un directorio especificado.

En la siguiente tabla se muestran los sistemas de archivos más comunes en Linux.

<u>TIPO</u>	<u>DESCRIPCIÓN</u>
ext3 ó reiserFS	Sistema de archivos de Linux.
msdos	Sistema de archivos de DOS.
vfat	Sistema de archivos de Windows 9X (nombres largos).
iso9660	Sistema de archivos de CD-ROM.
nfs	Sistema de archivos compartido por red ("exportado").

- **Desmontar:** decirle a Linux que se ha dejado de utilizar un determinado dispositivo.

Para *montar* un determinado sistema de archivos de un dispositivo, se utiliza el comando *mount*. La sintaxis es la siguiente: `# mount -t sistema_archivos dispositivo directorio [-o opciones]`, donde:

- *sistema\_archivos*, puede ser cualquiera de los que aparece en la tabla anterior.
- *dispositivo*, puede ser cualquier dispositivo del directorio `/dev` o, en el caso de *nfs*, un directorio de otro PC.
- *directorio*, es el directorio donde estará el contenido del dispositivo.
- *opciones* pueden ser cualquiera de la tabla siguiente, en el caso de no poner ninguna opción, *mount* utilizará las opciones por defecto

<u>OPCIÓN</u>	<u>DESCRIPCIÓN</u>
rw	Lectura/escritura.
ro	Sólo lectura.
exec	Se permite ejecución.
user	Los usuarios pueden "montar"/"desmontar".
suid	Tiene efecto los identificadores de propietario y del grupo.
auto	Se puede montar automáticamente.
async	Modo asíncrono.
sync	Modo síncrono.
dev	Supone que es un dispositivo de caracteres o bloques.

Una vez montado el dispositivo, si no se va a volver utilizar se puede desmontarlo con el comando *umount* con la siguiente sintaxis: `# umount directorio`

Siempre, después de utilizar un dispositivo hay que desmontarlo, para que se almacenen correctamente los datos en dicho dispositivo. Un ejemplo de ello, es el hecho de que, un lector de CD-ROM, que haya sido montado, no se abrirá hasta que no se desmonte.

Con un archivo `/etc/fstab`, cualquier usuario podría hacer: `$ mount /mnt/msdos+`

`$ umount /mnt/msdos+`

para montar y desmontar un disquete, respectivamente. Sin embargo, sólo el administrador podría montar y desmontar el directorio `/mnt/host2`.

Se muestran algunos ejemplos:

■ *Disquete de DOS*

```
mount -t msdos /dev/fd0 /mnt/floppy -o rw,noexec
```

```
umount /mnt/floppy
```

■ *Disquete de Windows 9X*

```
mount -t vfat /dev/fd0 /mnt/floppy -o user,rw
```

```
umount /mnt/floppy
```

■ *CD-ROM*

```
mount -t iso9660 /dev/cdrom /mnt/cdrom -o ro
```

```
umount /mnt/cdrom
```

*Directorio exportado de host2*

```
mount -t nfs host2:/tmp /mnt/host2
```

```
umount /mnt/host2
```

***Archivo `/etc/fstab`***

Lista los sistemas de archivos montados automáticamente en el arranque del sistema por el comando `mount -a` (en `/etc/rc` o archivo de inicio equivalente). En Linux, este archivo también contiene información acerca de áreas de swap utilizadas automáticamente por `swapon -a`.

*El primer campo*, (`fs_spec`), describe el dispositivo especial de bloque o sistema de archivos remoto a ser montado.

*El segundo campo*, (fs\_file), describe el punto de montaje para el sistema de archivos. Para particiones de intercambio (swap), este campo debe decir ``none".

*El tercer campo*, (fs\_vfstype), describe el tipo del sistema de archivos. Actualmente, el sistema soporta los siguientes tipos de sistemas de archivos. (y posiblemente otros - consulte /proc/filesystems): *minix*, *ext2*, *ext3*, *Raiserfs*, *msdos*, *hpfs*, *iso9660*, *nfs*, *swap*. Si vfstype tiene el valor *ignore*, la entrada es ignorada. Esto es útil para ver aquellas particiones del disco que no están siendo usadas.

*El cuarto campo*, (fs\_mntops), describe las opciones de montaje asociadas con el sistema de archivos. Es una lista de opciones separadas por comas. Contiene como mínimo el tipo de montaje y otras opciones apropiadas para el tipo del sistema de archivos. Las distintas opciones para sistemas de archivos locales están documentadas en mount(8). Las opciones específicas para nfs están documentadas en nfs(5). Las siguientes opciones son comunes para cualquier tipo de sistema de archivos: *noauto* (no monta el sistema cuando se ejecuta "mount -a", p.ej., cuando arranca el sistema), y *user* (permite que un usuario monte el sistema de archivos). Para mayor información.

*El quinto campo*, (fs\_freq), lo utiliza el comando dump(8) para determinar qué sistemas de archivos necesitan ser volcados (dumped). Si el quinto campo está vacío, dump asume que el sistema de archivos no necesita ser volcado.

*El sexto campo*, (fs\_passno), lo usa el programa fsck(8) para determinar el orden en el cual se van a chequear los sistemas de archivos cuando el sistema arranca. El sistema de archivos raíz debería llevar fs\_passno igual a 1, y otros sistemas de archivos deberían llevar fs\_passno igual a 2. Sistemas de archivos en un mismo disco serán comprobados secuencialmente, pero sistemas de archivos en diferentes discos serán comprobados al mismo tiempo para utilizar el paralelismo disponible en el equipo. Si el sexto campo no está presente o tiene un valor de 0, fsck asumirá que los sistemas de archivos no necesitan ser chequeados.

La forma apropiada de leer los registros de fstab es usando las rutinas getmntent

## **CUOTAS DE DISCO**

Una cuota de disco es la serie de parámetros y limitaciones que se asignan para el uso sobre ciertos recursos. Existen básicamente dos tipos de cuotas:

- Cuota de uso o de bloques: limita la cantidad de espacio en disco que puede ser utilizado.
- Cuota de archivos o inodos: limita la cantidad de archivos y carpetas que pueden ser creadas.

Asímismo, usualmente se establecen dos niveles en la administración de las cuotas:

- Cuota suave: este nivel se establece como un nivel de advertencia, en donde el usuario (o grupos) son informados de que están próximos a alcanzar su límite
- Cuota dura: este nivel es el límite de capacidad asignado al usuario o grupo, aún cuando se puede establecer un periodo de gracia (usualmente 7 días) en los que se pueden exceder estas cuotas

Las cuotas se definen por usuario o por grupos, y son controladas por el administrador del sistema. Estas cuotas se establecen para que los usuarios no hagan uso desmedido de los recursos que se les proporciona, optimizando estos y reduciendo los costos asociados al despilfarro de capacidad.

Los sistemas de archivos manejados por Linux y Unix generalmente soportan la implementación de cuotas de disco. Entre estos encontramos a ext2, ext3, reiserfs, ufs, etc. Los sistemas de archivos msdos (fat12, fat16, fat32) NO soportan la implementación de cuotas, no así ntfs.

Regularmente las cuotas se asignan por cada sistema de archivos, de manera independiente. Dependiendo del tipo de gestión que desee implementarse, se habilitarán o no las cuotas en distintos sistemas de archivos.

### ***Activando Cuotas***

Para habilitar el soporte para cuotas en un sistema de archivos determinado, deberemos asegurarnos de que ningún proceso esté haciendo uso de dicho recurso. Para asegurarnos de que este requisito sea cubierto, deberemos iniciar nuestro sistema en nivel de ejecución monousuario.

NOTA: Aún cuando podemos cambiarnos del actual nivel de ejecución hacia el nivel monousuario (init 1), esto no nos asegura que no haya algún proceso haciendo uso del recurso. Es por ello que se recomienda reiniciar el equipo e iniciarlo en dicho nivel de ejecución.

Por cada sistema de archivos al que se desee implementar la cuota, se deberán crear 4 archivos: quota.user, quota.group, aquota.user, aquota.group. Estos archivos serán los responsables de administrar las cuotas y llevar los índices de los archivos de los usuarios y grupos.

Suponiendo que tenemos 3 distintos sistemas de archivos a los que queremos aplicar las cuotas (/home, /var, /deptos), deberemos crear los 4 archivos arriba mencionados en cada punto de montaje: *# touch /home/quota.user,quota.group,aquota.user,aquota.group*

```
# touch /var/quota.user,quota.group,aquota.user,aquota.group
```

```
# touch /deptos/quota.user,quota.group,aquota.user,aquota.group
```

Si queremos optimizar la ejecución de dichos comandos, y se tiene un poco más de pericia sobre el manejo de bash, podremos resumir en una línea como sigue:

```
# touch /{home,var,deptos}/{,a}quota.{user.group}
```

En el archivo /etc/fstab, se debe especificar el montaje de dichos sistemas de archivos con las opciones de cuotas de usuario y/o cuotas de grupo:

- Cuota de uso o de bloques: limita la cantidad de espacio en disco que puede ser utilizado.
- Cuota de archivos o inodos: limita la cantidad de archivos y carpetas que pueden ser creadas.

Asímismo, usualmente se establecen dos niveles en la administración de las cuotas:

```
/dev/sda3           /home ext3 defaults,usrquota 1 1  
  
LABEL=/var         /var ext3 defaults,usrquota,grpquota 1 1  
  
/dev/mapper/VG01/LV01 /deptos ext3 defaults,grpquota 1 1
```

Debemos volver a montar dichos sistemas de archivos:

```
# mount -o remount /home  
  
# mount -o remount /var  
  
# mount -o remount /deptos
```

Una vez hecho esto, verificamos las opciones de montaje mediante el comando mount. Los archivos que generamos, en este momento no son mas que simples archivos en blanco, tenemos que convertirlos a un formato de cuotas:

```
# quotacheck -ugav
```

u : Activa las cuotas de usuarios

g : Activa las cuotas de grupos

a : Verifica la creación de cuotas en todos los sistemas de archivos con soporte para estas

v : Muestra una salida detallada de la ejecución del mandato. Es usual ver que el sistema nos envía un mensaje de advertencia cuando ejecutamos este mandato por primera vez, ya que se están generando los índices.

Activamos las cuotas en los puntos de montaje especificados:

```
# quotaon /home
```

```
# quotaon /var
```

```
# quotaon /deptos
```

Listo. Con esto, ya tenemos capacidad para la administración de cuotas de disco en nuestro sistema.

### ***Edición de Cuotas***

El mandato *edquota* nos permite editar las cuotas de disco que se implementarán. La sintáxis de este mandato es muy sencilla, solamente debemos enviarle como parámetro el nombre del usuario o del grupo del que requiramos:

```
# edquota paco
```

```
# edquota -g sistemas
```

Al ejecutar *edquota*, se ejecutará el editor de textos *vi* con opciones específicas para el manejo de cuotas. En la primer línea tenemos el identificador del usuario o grupo que estemos administrando. En las líneas subsecuentes, encontramos 7 columnas:

- *Filesystem*: sistema de archivos en el que se implementa la cuota.
- *blocks*: la actual cantidad de espacio en disco utilizado por el usuario o grupo en ese sistema de archivos. Este dato no se podrá modificar manualmente, aún cuando lo intentemos los cambios no se guardarán.
- *soft*: la cuota suave para la cantidad de espacio en disco utilizado. Se utiliza 0 para desactivar



esta cuota.

- *hard*: La cuota dura para la cantidad de espacio en disco a utilizar. El valor 0 desactiva esta cuota.
- *inodes*: La actual cantidad de archivos y carpetas utilizados por el usuario. Este dato no se podrá modificar manualmente, aún cuando lo intentemos los cambios no se guardarán
- *soft*: La cuota suave para la cantidad de archivos about:y carpetas utilizados. Se utiliza 0 para desactivar esta cuota.
- *hard*: La cuota dura para la cantidad de archivos y carpetas utilizados. El valor 0 desactiva esta cuota

### **# *edquota -t***

Este mandato nos permite establecer un periodo de gracia que acomode mejor a nuestras necesidades. Lo podemos establecer en días, horas, minutos o segundos, y se especificará para cada sistema de archivos.

## UNIDAD IX - FUNDAMENTOS DE REDES TCP/IP

### PROTOCOLO TCP/IP

La Familia de protocolos de internet es un conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de coputadoras. En ocasiones se la denomina *conjunto de protocolos TCP/IP*, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron los dos primeros en definirse, y que son los más utilizados de la familia. Existen tantos protocolos en este conjunto que llegan a ser más de 100 diferentes, entre ellos se encuentra el popular HTTP (HyperText Transfer Protocol), que es el que se utiliza para acceder a las páginas web, además de otros como el ARP (Address Resolution Protocol) para la resolución de direcciones, el FTP (File Transfer Protocol) para transferencia de archivos, y el SMTP (Simple Mail Transfer Protocol) y el POP (Post Office Protocol) para correo electrónico, TELNET para acceder a equipos remotos, entre otros.

El TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN). TCP/IP fue desarrollado y demostrado por primera vez en 1972 por el departamento de defensa de los Estados Unidos, ejecutándolo en ARPANET, una red de área extensa del departamento de defensa.

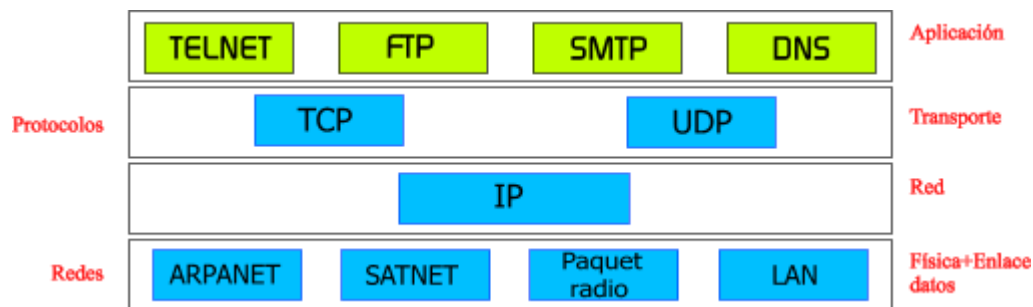
La familia de protocolos de internet puede describirse por analogía con el modelo OSI, que describe los niveles o capas de la pila de protocolos, aunque en la práctica no corresponde exactamente con el modelo en Internet. En una pila de protocolos, cada nivel soluciona una serie de problemas relacionados con la transmisión de datos, y proporciona un servicio bien definido a los niveles más altos. Los niveles superiores son los más cercanos al usuario y tratan con datos más abstractos, dejando a los niveles más bajos la labor de traducir los datos de forma que sean físicamente manipulables.

### PROTOCOLOS DE APLICACIONES

La pila TCP/IP incluye protocolos de aplicación tales como:

- TELNET para el acceso interactivo de una terminal a un host remoto.

- FTP ("File Transfer Protocol") para transferencias de alta velocidad de un disco a otro.
- SMTP ("simple mail transfer protocol") como sistema de correo de Internet.
- Estas son las aplicaciones implementadas más ampliamente, pero existen muchas otras. Cada implementación TCP/IP particular incluye un conjunto más o menos restringido de protocolos de aplicación.



Usan UDP o TCP como mecanismo de transporte. Recordar que UDP no es fiable ni ofrece control de flujo, por lo que en este caso la aplicación ha de proporcionar sus propias rutinas de recuperación de errores y de control de flujo. Suele ser más fácil desarrollar aplicaciones sobre TCP, un protocolo fiable, orientado a conexión. La mayoría de los protocolos de aplicación utilizan TCP, pero algunas aplicaciones se construyen sobre UDP para proporcionar un mejor rendimiento reduciendo la carga del sistema que genera el protocolo. La mayoría de ellas usa el modelo de interacción cliente/servidor.

## DIRECCIONES PRIVADAS

Tradicionalmente, las redes IP fueron agrupadas en clases cuyas direcciones de red componentes eran de 8, 16 o 24 bits de tamaño.

	direcciones IP		máscara de red	longitud
<b>Clase A</b>	1.0.0.0	- 126.255.255.255	255.0.0.0	= /8
<b>Clase B</b>	128.0.0.0	- 191.255.255.255	255.255.0.0	= /16
<b>Clase C</b>	192.0.0.0	- 223.255.255.255	255.255.255.0	= /24

Las direcciones IP que no se encuentran en estos rangos se utilizan para propósitos especiales.

En cada clase existen rangos de direcciones reservados para su uso en redes de área local (LANs). Se

garantiza que estas direcciones no entren en conflicto con las direcciones propias de Internet (en consecuencia, si una de estas direcciones se asigna a un equipo éste no podrá acceder a Internet directamente sino a través de una puerta de enlace que actúe como proxy para los servicios individuales o hacer la traducción de direcciones de red – NAT) Estos rangos de direcciones se dan en la siguiente tabla junto con el número de rangos en cada clase.

	<b>direcciones de red</b>	<b>longitud</b>	<b>cantidad</b>
<b>Clase A</b>	10.x.x.x	/8	1
<b>Clase B</b>	172.16.x.x - 172.31.x.x	/16	16
<b>Clase C</b>	192.168.0.x - 192.168.255.x	/24	256

La primera dirección en una red IP es la dirección de la propia red. La última dirección es la dirección de difusión de la red. Todas las otras direcciones se pueden asignar a máquinas de la red. De éstas, la primera o la última dirección generalmente se asigna a la puerta de enlace para Internet.

## **CONFIGURACIÓN DE LA RED**

Las herramientas tradicionales de configuración de red a bajo nivel en sistemas GNU/Linux son los programas `ifconfig` y `route` que vienen en el paquete `net-tools`. Estas herramientas han sido oficialmente reemplazadas por `ip` que viene en el paquete `iproute`. El programa `ip` funciona con Linux 2.2 y superior y es más poderoso que las herramientas anteriores. Sin embargo, las herramientas anteriores aún funcionan y resultan más familiares para muchos usuarios.

### ***Configuración de la red a bajo nivel – ifconfig y route***

Veamos una ilustración de cómo cambiar la dirección IP de la interfaz `eth0` de 192.168.0.3 a 192.168.0.111 y convertir a `eth0` en la ruta a la red 10.0.0.0 vía 192.168.0.1. Empiece ejecutando `ifconfig` y `route` sin argumentos para mostrar el estado actual de todas las interfaces de red y encaminamiento.

#### **# ifconfig**

```
eth0 Link encap:Ethernet      HWaddr 08:00:46:7A:02:B
```

```
inet addr:192.168.0.3 Bcast:192.168.0.255 Mask:255.255.255.0
```

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:23363 errors:0 dropped:0 overruns:0 frame:0

TX packets:21798 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:100

RX bytes:13479541 (12.8 MiB) TX bytes:20262643 (19.3 MiB)

Interrupt:9

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

UP LOOPBACK RUNNING MTU:16436 Metric:1

RX packets:230172 errors:0 dropped:0 overruns:0 frame:0

TX packets:230172 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:22685256 (21.6 MiB) TX bytes:22685256 (21.6 MiB)

## **# route**

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0 *		255.255.0.0	U	0	0	0	eth0
default	192.168.0.1	255.255.255.255	UG	0	0	0	eth0

Primero deshabilitamos la interfaz.

**# ifconfig eth0 inet down**

**# ifconfig**

lo Link encap:Local Loopback

... (no más entradas eth0)

**# route**

... (no más entradas en la tabla de rutas)

Luego la habilitamos con la nueva IP y la nueva ruta.

```
# ifconfig eth0 inet up 192.168.0.111 \
```

```
netmask 255.255.0.0 broadcast 192.168.255.255
```

```
# route add -net 10.0.0.0 netmask 255.0.0.0 gw 192.168.0.1 dev eth0
```

El resultado:

```
# ifconfig
```

```
eth0 Link encap:Ethernet HWaddr 08:00:46:7A:02:B0
```

```
inet addr:192.168.0.111 Bcast:192.168.255.255 Mask:255.255.0.0
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

```
...
```

```
lo Link encap:Local Loopback
```

```
inet addr:127.0.0.1 Mask:255.0.0.0
```

```
...
```

```
# route
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0 *		255.255.0.0	U	0	0	0	eth0
10.0.0.0	192.168.0.1	255.0.0.0	UG	0	0	0	eth0

***Configuración de la red a bajo nivel – ip***

Los comandos ip equivalentes a los comandos ifconfig y route anteriores son:

```
ip link show

ip route list

ip link set eth0 down

ip addr del dev eth0 local 192.168.0.3

ip addr add dev eth0 local 192.168.0.111/16 broadcast
192.168.255.255

ip link set eth0 up

ip route add dev eth0 to 10.0.0.0/8 src 192.168.0.111 via
192.168.0.1
```

El programa ip muestra la sintaxis de sus comandos cuando se ejecuta con el argumento help. Por ejemplo, ip link help imprime por pantalla:

```
Usage: ip link set DEVICE { up | down | arp { on | off } |
        dynamic { on | off } |
        multicast { on | off } | txqueuelen PACKETS |
        name NEWNAME |
        address LLADDR | broadcast LLADDR |
        mtu MTU }

ip link show [ DEVICE ]
```

### ***Configurando una interfaz***

Para interfaces Wi-Fi se utiliza el programa iwconfig, que viene con el paquete wireless-tools, además de ifconfig o ip.

### ***Configurar interfases de red***

A fin de facilitar la configuración de la red, Debian proporciona una herramienta estándar de configuración de red de alto nivel que consiste en los programas ifup, ifdown y el archivo /etc/network/interfaces. 9 Si elige utilizar ifupdown para realizar la configuración de su red, entonces no debería usar los comandos de bajo nivel. 10 Ifupdown se programó bajo la suposición que sólo iba a ser utilizado para configurar y desconfigurar las interfaces de red.

Para actualizar la configuración de la interfaz haga lo siguiente:

```
# ifdown eth0

# editor /etc/network/interfaces      # modifique a su antojo

# ifup eth0
```

### ***Configurando una interfaz con una dirección IP estática***

Supongamos que desea configurar una interfaz Ethernet que tiene una dirección IP fija 192.168.0.123. Esta dirección comienza con 192.168.0 por lo tanto debe estar en una LAN. Supongamos además que 192.168.0.1 es la dirección de la puerta de enlace de la LAN a Internet. Edite /etc/network/interfaces de modo que incluya un fragmento como el siguiente:

```
iface eth0 inet    static

    address  192.168.0.123

    netmask  255.255.255.0

    gateway  192.168.0.1
```

Si tiene instalado el paquete resolvconf puede añadir líneas para especificar la información relativa al DNS. Por ejemplo:

```
iface eth0 inet static

    address 192.168.0.123

    netmask 255.255.255.0

    gateway 192.168.0.1

    dns-search nicedomain.org
```



```
dns-nameservers 195.238.2.21 195.238.2.22
```

Luego que se activa la interfaz, los argumentos de las opciones dns-search y dns-nameservers quedan disponibles para resolvconf para su inclusión en resolv.conf. El argumento lindodominio.org de la opción dns-search corresponde al argumento de la opción search en resolv.conf(5). Los argumentos 195.238.2.21 y 195.238.2.22 de la opción dns-nameservers corresponde a los argumentos de las opciones nameserver en resolv.conf(5). Otras opciones reconocidas son dns-domain y dns-sortlist.

### ***Configurando una interfaz usando DHCP***

Para configurar una interfaz usando DHCP edite el /etc/network/interfaces de manera que incluya un fragmento como el siguiente :

```
iface eth0 inet dhcp
```

Para que esto funcione debe tener instalado uno de los clientes DHCP mencionados anteriormente.

### **Configurando una interfaz Wi-Fi**

El paquete wireless-tools incluye el script /etc/network/if-pre-up.d/wireless-tools que permite configurar hardware Wi-Fi (802.11a/b/g) antes que se active la interfaz.

Para cada parámetro posible del comando iwconfig puede incluir una opción en /etc/network/interfaces con un nombre como el del parámetro con el prefijo “wireless-”. Por ejemplo, para fijar el ESSID de eth0 en miessid y la clave de cifrado en 123456789e antes de activar eth0 usando DHCP, edite el /etc/network/interfaces de modo que incluya un fragmento como el siguiente :

```
iface eth0 inet dhcp
```

```
wireless-essid miessid
```

```
wireless-key 123456789e
```

Obsérvese que no debería utilizar este método para configurar el ESSID y la clave si está ejecutando waproamd para esta interfaz. En el momento que se ejecuta ifup, waproamd ya tiene configurados ESSID y la clave.

### ***Configurando múltiples interfaces Ethernet para una puerta de enlace***

Supongamos que eth0 está conectada a Internet con una dirección IP configurada con DHCP y eth1 está

conectada a la LAN con una dirección IP estática 192.168.1.1. Edite `/etc/network/interfaces` de modo que incluya un fragmento similar al siguiente:

```
iface eth0 inet dhcp

iface eth1 inet static

address 192.168.1.1

netmask 255.255.255.0
```

Si activa NAT en esta máquina, puede compartir la conexión de Internet con todas las máquinas de la LAN.

### ***Configurando interfases virtuales***

Usando interfaces virtuales puede configurar una única tarjeta Ethernet para que sea la interfaz de distintas subredes IP. Por ejemplo, supongamos que su máquina se encuentra en una red LAN 192.168.0.x/24. Desea conectar la máquina a Internet usando una dirección IP pública proporcionada con DHCP usando su tarjeta Ethernet existente. Edite `/etc/network/interfaces` de modo que incluya un fragmento similar al siguiente:

```
iface eth0 inet static

address 192.168.0.1

netmask 255.255.255.0

network 192.168.0.0

broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

*La interfaz eth0:0 es una interfaz virtual. Al activarse también lo hará su padre eth0.*

## **SECURE SHELL – SSH**

SSH (Secure SHell) es la manera segura de comunicarse a través de Internet. Una versión libre de SSH llamada OpenSSH se encuentra disponible en el paquete Debian `ssh`.

## ***Fundamentos***

Primero instale el cliente y el servidor OpenSSH. *# apt-get update && apt-get install ssh.*

*/etc/ssh/sshd\_not\_to\_be\_run* no debe estar presente si desea ejecutar el servidor

## ***OpenSSH.***

SSH tiene dos protocolos de autenticación:

- *Protocolo SSH versión 1:* la versión que viene con Potato admite únicamente este protocolo. Métodos de autenticación disponibles: RSA Authentication (autenticación del usuario basada en una clave RSA), Rhosts Authentication (autenticación basada en *.rhosts* - insegura, desactivada), Rhosts RSA Authentication ( autenticación basada en *.rhosts* combinada con una clave RSA – desactivada), Challenge Response Authentication: autenticación basada en challenge-response. RSA Password Authentication (autenticación basada en contraseña).
- *Protocolo SSH versión 2:* versiones posteriores a Woody usan este protocolo como protocolo principal.

## ***Métodos de Autenticación Disponibles***

- PubkeyAuthentication: autenticación del usuario basada en una clave pública
- Hostbased Authentication: autenticación basada en *.rhosts* o */etc/hosts.equiv* combinada con la autenticación de la clave pública de la máquina cliente (desactivada).
- Password Authentication: autenticación basada en contraseña.
- Challenge Response Authentication: autenticación basada en challenge response.

*/etc/ssh/ssh\_config* es el archivo de configuración más importante, las entradas son:

- *Host:* Restringe las siguientes declaraciones (up to the next Host keyword) siendo nada mas los host que declaremos aquí los que se les dará una llave de autenticación ssh para entablar comunicación.
- *Protocol:* especifica la versión del protocolo SSH. Valor predeterminado “2,1”.
- *Preferred Authentications:* especifica el método de autenticación para el cliente SSH2. Por

defecto: “hostbased,publickey,keyboard-interactive,password”.

- *ForwardX11*: desactivado por defecto. Se puede no tener en cuenta mediante la opción “-X” de la línea de comandos

Otro archivo de interés: */etc/ssh/sshd\_config*: valores predeterminados del servidor SSH. Las entradas más importantes son:

- *ListenAddress*: especifica las direcciones locales que sshd debe escuchar. Se permiten múltiples opciones.
- *AllowTcpForwarding*: desactivado por defecto.
- *X11Forwarding*: desactivado por defecto.

*\$HOME/.ssh/authorized\_keys*: la lista de las claves públicas predeterminadas que los clientes usan para conectarse con la cuenta en este host.

## COMANDOS SSH Y SCP

Lo siguiente iniciará una conexión ssh desde un cliente:

```
$ ssh nombre_usuario@nombre_máquina.dominio.ext
```

```
$ ssh -l nombre_usuario@nombre_máquina.dominio.ext # Fuerza la versión 1 de SSH
```

```
$ ssh -l -o RSAAuthentication=no -l username foo.host
```

```
# force password on SSH1
```

```
$ ssh -o PreferredAuthentications=password -l username foo.host
```

```
# force password on SSH2
```

Para el usuario, las funciones de ssh son mejores y más seguras que las de telnet. Se pueden copiar archivos hacia y desde un host remoto usando el programa scp. Su sintaxis es similar al convencional cp con la excepción que debe especificar un nombre de host antes del archivo, significando que el camino del archivo está en el host especificado. El siguiente ejemplo ilustra la sintaxis de scp copiando un archivo local llamado */tmp/fred* al */home/maggie/* del host remoto *chianti.vbrew.com*:

```
$ scp /tmp/fred vchianti.vbrew.com:/home/maggie/
```

```
maggie@vchianti.vbrew.com's password:
```

```
fred      100% |*****| 50165      00:01 ETA
```

De nuevo, se le pedirá una clave. La orden scp muestra el progreso de la copia por omisión. Puede copiar un archivo desde un host remoto con la misma facilidad; simplemente especificando su nombre de host y ruta como origen y la ruta local como destino. También se puede copiar un archivo desde un host remoto a otro host remoto, pero habitualmente no necesitará hacer eso, porque todos los datos viajan a través de su host.

## SERVICIO VNC

Hay muchos servidores VNC, pero explicaremos el uso de *x11vnc* porque es el más sencillo de utilizar. La mayoría de servidores VNC requieren un display de las X particular y, aunque ofrecen un escritorio remoto, lo que hacen es iniciar una nueva sesión gráfica en vez de ofrecer acceso a una sesión ya existente. Con *x11vnc* podremos permitir el acceso a una sesión X ya existente de una forma sencilla.

### *Instalación de VNC*

Instalamos el paquete para el Servidor: `# apt-get install x11vnc`

### *Arrancando el servidor*

Para arrancar el servidor, abriremos una consola y escribiremos el comando *x11vnc*. Esto nos iniciará un servidor básico, sin contraseña, que permite el acceso a todo el mundo y que una vez ha desconectado el cliente, se cierra.

Veremos ahora qué parámetros podemos pasarle al inicio, para configurar el servidor de una forma más razonable:

- *bg*: Nos inicia el servidor en segundo plano. Para poder cerrar la consola y que siga en marcha.
- *passwd*: Establece la contraseña que se pedirá a los clientes al conectar.
- *gui*: Inicia el interfaz gráfico (un poco precario) del servidor.

Sabiendo estos parámetros, podríamos iniciar el servidor de VNC de esta manera, para que nos

aparezca su ventana de configuración:*x11vnc -bg -gui -passwd mi\_contraseña*, se nos abrirá la pantalla de configuración, en la que podremos configurar las opciones del servidor. Ahora, abrimos otra consola (en el equipo Cliente) y pasamos a instalar el Paquete para el Cliente, siendo ROOT *# apt-get install xtightvncviewer*.

Para iniciar Sesión en el Servidor, desde la Consola cliente tecleamos: *\$xtightvncviewer*.

## UNIDAD X - SERVICIO CUPS

### FUNCIONAMIENTO DE CUPS

El Sistema de impresión común de Unix (Common Unix Printing System en inglés, abreviado CUPS) es un sistema de impresión modular para sistemas operativos de tipo Unix que permite que un computador actúe como servidor de impresión. Un computador que ejecuta CUPS actúa como un servidor que puede aceptar tareas de impresión desde otros computadores clientes, los procesa y los envía al servidor de impresión apropiado.

CUPS está compuesto por una cola de impresión con su planificador, un sistema de filtros que convierte datos para imprimir hacia formatos que la impresora conozca, y un sistema de soporte que envía los datos al dispositivo de impresión. CUPS utiliza el protocolo IPP(Internet Printing Protocol) como base para el manejo de tareas de impresión y de colas de impresión. También provee los comandos tradicionales de línea de comandos de impresión de los sistemas Unix, junto a un soporte limitado de operaciones bajo el protocolo server message block (SMB).

Los controladores de dispositivos de impresión que CUPS provee pueden ser configurados utilizando archivos de texto con formato Descripción de impresoras PostScript (PPD, PostScript Printer Description en inglés) de Adobe Systems. Existen varias interfaces de usuario para diferentes plataformas para configurar CUPS; cuenta también con una interfaz como aplicación Web. CUPS es software libre y se distribuye bajo licencia GNU General Public License y GNU Lesser General Public License, Versión 2.

#### ***Realizar la Instalación de CUPS.***

```
#apt-get install cupsys
```

#### ***Configurar el servicio CUPS.***

Filtros, controladores, PPDs ... El flujo de impresión se compone de varios elementos que debemos conocer para entender bien cómo encajan todas las piezas. El proceso general es el siguiente: la aplicación suministra un archivo en Postscript, PDF, texto plano o mapa de bits (JPG, PNG, etc.). El contenido de este archivo va al sistema de impresión, que mira a qué impresora tiene que enviarlo. Una

vez decidido eso, busca las características de la impresora para saber si tiene que hacer alguna transformación: hay impresoras que pueden recibir Postscript sin traducción, y en esos casos el sistema de impresión solo pasa lo que a su vez le proporcionó la aplicación, convirtiéndolo a Postscript antes si no viniera ya en ese formato. Como lo normal (para un usuario doméstico) es que la impresora no “hable” Postscript, el sistema de impresión tiene que buscar una forma de convertir los datos al lenguaje nativo de ésta. Los programas encargados de esto son los filtros. Estos filtros pueden recibir como entrada los datos de la aplicación, o necesitar un paso intermedio en el que se convierten a una representación gráfica llamada raster.

Asimismo, los filtros pueden hacer otras cosas aparte de convertir un archivo raster a lenguaje nativo de la impresora: pueden reducir el tipo de letra, poner varias páginas “virtuales” por página “real”, etc. Estos filtros conocen los lenguajes nativos de las impresoras por varios controladores que tienen configurados.

Nuestro sistema de impresión será CUPS. Como filtros podemos usar, aparte de algunos filtros que trae CUPS de serie (y que podemos ver en `/usr/lib/cups/` filter), Ghostscript, Foomatic o los filtros que proporcione el fabricante de nuestra impresora. Ghostscript trae controladores para muchas impresoras, y Foomatic también; pero también podemos usar controladores externos, como los proporcionados por Gutenprint.

## **PPDS**

Otro elemento necesario para configurar una impresora en CUPS es un archivo PPD. Un archivo PPD (Postscript Printer Description) describe las características de una impresora. Estos archivos solo sirven para impresoras Postscript, pero CUPS ha extendido el concepto y también usa archivos PPD para impresoras no Postscript. En ellos incluye las llamadas a los filtros que convertirán el código Postscript de entrada en el formato nativo de la impresora. Están, por lo tanto, un nivel por encima de los filtros.

Todas las impresoras Postscript traerán su archivo PPD correspondiente en el CD de instalación, o estará disponible en la página web del fabricante. Para las impresoras no Postscript, cientos de PPDs han sido creados por los desarrolladores de CUPS y muchos otros voluntarios.

Foomatic, una “suite” de la que hablamos más adelante, ofrece una colección de PPDs con soporte para decenas de impresoras.



### ***Configurar la Impresora Predeterminada.***

La dificultad de instalación de una impresora en CUPS depende de varios factores. Las impresoras Postscript suelen ser las más fáciles de configurar, mientras que las impresoras “caseras” de tinta pueden dar problemas.

Algunas impresoras de bajo coste hacen parte de sus funciones por software, y necesitan un controlador especial que no está disponible para Linux. En cualquier caso, los pasos a seguir son siempre los mismos: decir a CUPS cuál es el interfaz a usar, y darle una forma de escoger “cómo hablar” con la impresora.

CUPS viene de serie con soporte para varias familias de impresoras, pero podemos hacer que soporte cualquier otra siempre que suministremos los PPDs y filtros adecuados.

### ***Configuraciones***

Hay dos formas de configurar CUPS. La primera es editando el archivo de configuración */etc/cups/cupsd.conf*. La segunda es a través de una interfaz web clásica, ubicada en el puerto 631. Ésta es muy parecida a la configuración web de muchas impresoras aptas para redes, y debería ser intuitiva para todos.

## **LA INTERFAZ WEB DE ADMINISTRACIÓN DE CUPS**

Todas las labores de administración de CUPS las haremos a través de su interfaz web. A este interfaz se puede acceder lanzando el navegador desde el mismo equipo en el que está CUPS, y apuntándolo a <http://localhost:631> (por defecto el interfaz solo funciona en localhost). Necesitamos usuario y contraseña para entrar en este interfaz, que por defecto serán “root” y la contraseña de root del equipo.



### **Interfaz web de administración de CUPS.**

De momento solo nos interesa saber cómo instalar una impresora, y por suerte en cuanto entremos en el interfaz veremos un botón titulado “Añadir impresora” y que vale justo para eso.

### **IMPRESORAS LOCALES (USB Y PARALELO)**

Un primer paso importante pero que quizás se nos olvide es encender la impresora. En muchos casos, CUPS puede detectar la impresora que está conectada en local, y nos la ofrecerá como opción cuando empecemos a instalarla. Vamos a suponer que éste es el caso, porque es lo normal con las impresoras USB. Las impresoras por puerto paralelo son, ahora mismo, una reliquia.

Los primeros datos que tenemos que rellenar sobre una impresora son el nombre, la ubicación y una descripción. La ubicación y la descripción pueden ser cualquier cosa, y son solo datos “cosméticos” para tener una idea más clara de cuál es la impresora (por si tuviéramos muchas).

El nombre sí que es un dato importante: el nombre que demos en CUPS a la impresora será usado por los programas que quieran imprimir en ella, es el identificador único de la impresora en el sistema.

Como nos dice el comentario sobre la casilla que veremos en el interfaz, puede tener cualquier caracter

menos espacios, “#” y “/”. Por ejemplo: para una impresora Samsung ML-2010 (una impresora USB, láser y monocromo, que hemos usado en nuestras pruebas), podríamos usar nombres como “ml2010”, “ml-2010”, “samsungml2010”, etc. Tras rellenar estos datos nos aparece un menú desplegable marcado como “Conexión”. Nuestra impresora USB aparecerá en esa lista, marcada posiblemente con una etiqueta tipo “USB #1” o similar, en la que se indique el puerto USB que está ocupando.

La siguiente sección que tenemos que atender es el controlador de la impresora. Por defecto, CUPS trae unos cuantos, pero no son muchos. Si no está en la lista el que necesitamos, podemos proporcionar un archivo PPD con las características de la impresora. Este archivo podemos haberlo obtenido en el CD de controladores de la impresora, podemos haberlo descargado desde Internet, o podemos haberlo creado con alguna utilidad, como Foomatic, que veremos más adelante.

Con esto ya habremos instalado la impresora. Sin embargo, antes de llevarnos a la lista de impresoras, el proceso de instalación nos llevará a la página de opciones de la impresora. En esta sección es donde escogeremos cosas como el tipo de papel, copias por archivo que se mande a la impresora, etiquetas, etc. Las opciones concretas dependerán de cada impresora y del PPD que usemos. Ante la duda, lo razonable es usar las opciones que vienen por defecto.

La nueva impresora aparecerá en la pestaña “Impresoras” del interfaz web de CUPS. En los datos de la impresora aparecerán los que ya habíamos introducido nosotros, y además también el estado de la impresora (que debería ser “inactiva, aceptando trabajos, pública”) y la “URI” de la conexión. La URI de la impresora es el dispositivo al que está conectada, o su ubicación en la red si es una impresora remota. En el caso de la impresora Samsung ML-2010 USB que hemos usado en las pruebas, aquí aparecerá `usb://Samsung/ML-2010`.

En otras impresoras, la URI cambiará para reflejar marca y modelo de forma parecida. Cuando el PPD es obtenido del fabricante, es posible que éste también haya incluido algún filtro al que se hace referencia en el PPD. Si es así, veremos que cuando vayamos a la lista de impresoras aparecerá un error. Generalmente, el error será del tipo de “archivo no encontrado”, haciendo referencia al nombre del filtro que aparece en el PPD pero no está en la instalación de CUPS. No suele ser difícil buscar ese archivo en el CD del fabricante y copiarlo a `/usr/lib/cups/filter`, reiniciando luego CUPS para que lea otra vez la lista de filtros y lo encuentre. Si esto no funciona, puede que haya algún otro problema. Lo recomendable en este caso es mirar el log de errores de CUPS, en `/var/log/cups/error_log` (también podemos acceder desde la pestaña “Administración”, opción “Ver archivo deregistro de errores”). Los

mensajes que aparezcan ahí nos darán una pista de cuál es el problema.



## UNIDAD XI: SERVICIO NFS

### DEFINICIÓN DE NFS

El Sistema de archivos de Red (NFS, por sus siglas en inglés) es probablemente el servicio de red más prominente que usa RPC. Permite acceder a archivos en anfitriones remotos exactamente en la misma manera que se accedería si fueran locales. Una mezcla de soporte en el núcleo y demonios en espacio de usuario en el lado del cliente, junto con un servidor NFS en el lado del servidor, hace esto posible. Este acceso a los archivos es completamente transparente al cliente y funciona con varias clases de servidores y arquitecturas anfitrionas.

### CARACTERÍSTICAS ÚTILES DE NFS

NFS ofrece varias características útiles: Los datos accedidos por todos los usuarios pueden mantenerse en un anfitrión central, con los clientes montando este directorio en tiempo de arranque. Por ejemplo, se puede mantener todas las cuentas de usuario en un anfitrión y hacer que todos los anfitriones de la red monten el directorio /home desde ese anfitrión. Si se instala NFS junto a NIS, los usuarios pueden entrar en cualquier sistema y trabajar en un conjunto de archivos.

La información que consume grandes cantidades de disco puede mantenerse en un único anfitrión. Por ejemplo, todos los archivos y programas relativos a LaTeX y METAFONT pueden almacenarse y mantenerse en un lugar..

Los datos administrativos pueden almacenarse en un único anfitrión. No hay necesidad de usar **rcp** para instalar el mismo archivo estúpido en 20 máquinas diferentes. No es demasiado difícil preparar el funcionamiento básico de NFS en el cliente y el servidor

### EL SERVIDOR NFS

NFS funciona de la siguiente manera: primero, un cliente intenta montar un directorio de un anfitrión remoto en un directorio local justo de la misma manera que si fuese un dispositivo físico. Sin embargo, la sintaxis usada para especificar el directorio remoto es diferente. Por ejemplo, para montar /home desde el anfitrión *vlager* en /users en *vale*, el administrador escribe la siguiente orden: `# mount -t nfs`

*vlager:/home /users*

*mount* tratará de conectar con el demonio remoto sobre *rpc.mountd* de *vlager* vía RPC. El servidor verificará si vale tiene permiso para montar el directorio en cuestión, en cuyo caso, devuelve un descriptor de archivo.

Este descriptor será usado en todas las peticiones subsecuentes que se hagan sobre los archivos bajo *users*. Cuando alguien accede a un archivo sobre NFS, el núcleo manda una llamada de RPC a *rpc.nfsd* (el demonio de NFS) en la máquina servidor. Esta llamada toma el descriptor de archivo, el nombre del archivo a acceder y los identificadores de usuario y grupo del usuario como parámetros. Éstos se usan en la determinación de los derechos de acceso al archivo especificado. Para prevenir que usuarios no autorizados lean o modifiquen archivos, los identificadores de usuario y grupo deben ser iguales en ambos anfitriones... En la mayoría de las implementaciones de Unix, la funcionalidad NFS de cliente y servidor se implementan como demonios a nivel de núcleo que arrancan desde el espacio de usuario al arrancar la máquina. Éstos son los *Demonios NFS (rpc.nfsd)* en el anfitrión servidor, y *Block I/O Daemon (biod)* en el anfitrión cliente.

Para mejorar el rendimiento, *biod* realiza la E/S usando prelectura y postescritura asíncrona; también, varios demonios *rpc.nfsd* usualmente se ejecutan concurrentemente. La implementación actual de NFS de Linux es un poco diferente del NFS clásico en la que el código de servidor se ejecuta enteramente en espacio de usuario, así que ejecutar múltiples copias simultáneamente es más complicado. La implementación actual *derpc.nfsd* ofrece una característica experimental que permite soporte limitado para múltiples servidores. Olaf Kirch desarrolló el soporte para servidor NFS basado en el núcleo ofrecido en la versión.

### ***El servidor***

La primera cosa a hacer, como ya hemos visto, es iniciar portmap ya que este protocolo es necesario para NFS .

```
root >>/usr/sbin/rpcinfo -p
```

```
rpcinfo: can't contact portmapper: RPC: Remote system error - Connection refused
```

```
root >>/sbin/portmap
```

```
root >>/usr/sbin/rpcinfo -p
```

```
program vers proto port
```

***100000 2 tcp 111 portmapper***

***100000 2 udp 111 portmapper***

El comando `rpcinfo` muestra los servicios RPCs en la máquina especificada como argumento (opción `-p`). Notamos que `portmap` todavía no está funcionando: lo iniciamos (la mayoría de las distribuciones Linux proveen scripts para automatizar esto en el arranque) y comprobamos que funciona. Otra razón común para que `rpcinfo` responda negativamente es que el `portmapper` no permita la respuesta a causa de la restricción de seguridad en los archivos `/etc/hosts.{allow, deny}`. En este caso, añade una entrada `"portmap: hosts"` en el archivo `hosts.allow`.

Antes de que NFS se inicie por sí mismo, debe ser configurado. Existe un único archivo de configuración que se llama `/etc/exports`. Cada línea muestra la ruta exportada seguido de una lista de clientes a los que se permite el acceso. Se pueden añadir opciones al final de cada nombre de cliente. La página de manual `exports` (`man exports`) explica la sintaxis para los nombres de cliente y las opciones.

Se aceptan como nombres de cliente: nombre de la máquina, caracteres comodín en un nombre de dominio (v.gr. : `linux-*.mondomaine.fr`), un *netgroup* ( `@grupo`) si se usa NIS, una dirección IP, entre otras.

No vamos a detallar aquí todas las opciones de montaje disponibles, pero algunas de las más importantes son:

- `rw` (lectura/escritura): el cliente puede leer y escribir en el sistema exportado.
- `ro` (sólo lectura): el cliente sólo puede leer el sistema exportado.
- `root_squash` : es preferible que un usuario *root* del cliente no pueda escribir con permisos de *root*. Para impedirlo, UID/GID 0 (i.e. *root*) en el lado del cliente se traduce en el usuario *nobody*. Esta opción está activada por defecto, pero se puede cancelar con `no_root_squash`.
- `all_squash` : todos los clientes que acceden al sistema exportado utilizan el UID/GID de *nobody*.
- `anonuid, anongid`: el usuario *nobody* ahora usa los UID y GID definidos por estas opciones.

Ahora tenemos que iniciar los demonios `rpc.mountd` y `rpc.nfs` para tener funcionando el servidor NFS. Comprobamos nuevamente que todo está funcionando con el comando `rpcinfo`. Incluso podemos

inicializar el servidor para los protocolos *nsm* y *nlm* (*rpc.statd* y *rpc.lockd*, respectivamente). No hay ninguna premisa para arrancar un servidor NFS... pero es altamente recomendable que se reinicie por sí mismo, en caso de que la máquina falle, entre otras.

Cuando modificamos el archivo de configuración */etc/exports*, debemos avisar a los demonios implicados que se deben hacer los cambios. El comando *exportfs* transmite esta información a nuestros servidores. La opción *-r* sincroniza el archivo */etc/mtab*<sup>2</sup> con el archivo */etc/exports* file. La opción *-v* muestra juntos todos los sistemas de archivos exportados junto con sus opciones. Después de ponerse en marcha el servidor NFS, los siguientes archivos contienen información importante:

- */var/lib/nfs/rmtab* : cada línea muestra el nombre del cliente y el sistema de archivos importado desde este servidor.
- */var/lib/nfs/etab*: el archivo */etc/exports* sólo contiene una lista de peticiones. *etab* está creado por *exportfs*. Contiene en cada línea información detallada sobre las opciones usadas cuando se exporta un sistema de archivos a un solo cliente. Es el archivo de referencia usado por *rpc.mountd* cuando es arrancado
- */proc/fs/nfs/exports* contiene la lista de clientes conocida por el núcleo.
- */var/lib/nfs/xtab*: se usa por precisión cuando *etab* contiene nombres de clientes y grupos de máquinas con comodines. Este archivo sólo contiene nombres explícitos de máquinas.

Cuando un cliente quiere acceder a un sistema de archivos, empieza haciendo una petición *mountd*. Entonces se busca en *etab* si la petición está disponible. Se comprueba el núcleo para saber si el cliente tiene permitida la petición (comprobando *hosts.{allow, deny}*, reglas de cortafuegos, ...). El núcleo utiliza *exportfs* para la comprobación, permitiendo actualizar el archivo */var/lib/nfs/etab*. Si, en este archivo, el sistema exportado tiene permitido ser exportado al grupo al que pertenece el cliente, entonces *mountd* informa al núcleo que actualice *xtab* con este nuevo host.

## EL CLIENTE NFS

El acceso al sistema de archivos exportado por NFS está controlado directamente por el núcleo. Éste tiene que haber sido compilado para soportar NFS. El archivo */proc/filesystems* contiene una lista con todos los sistemas de archivos soportados directamente por el núcleo. Entonces, lo único que tiene que



hacer es decir al núcleo que quiere acceder a un sistema exportado por NFS.

El comando `mount` permite acceder a diferentes sistemas de archivos. Informa al núcleo que está disponible un nuevo sistema de archivos indicando su tipo, su *dispositivo* y su punto de montaje. Se puede usar la opción `-t` para indicar el tipo del sistema de archivos a usar. Para NFS, escribimos: `-t nfs`.

`mount` tiene sus propias opciones para NFS. Por ejemplo, se pueden utilizar las opciones `rsize` y `wsize` para cambiar el tamaño de los bloques para lectura o escritura. Puede combinar opciones específicas de NFS con opciones más generales como `intr`, `noexec` o `nosuid`. La página de manual `mount` muestra todas esas opciones.

Supongamos que la máquina *charly* tiene un servidor NFS y exporta su directorio `/usr/local`. Cuando quiera acceder desde la máquina *jill*, tendrá que montar el directorio exportado de *charly* a *jill*:  
`root@jill >> mount -t nfs -o nosuid,hard,intr charly:/usr/local /usr/local`

El comando indica que estamos montando un sistema de archivos NFS (`-t nfs`), con las opciones `nosuid`, `hard` e `intr`. Los dos últimos argumentos son los más interesantes. El primero de ellos especifica el *dispositivo* a montar.

Para NFS la sintaxis es distinta de la línea `mount` habitual, donde se especifica dispositivo y directorio. Aquí especificamos `servidor:directorio_exportado` en vez de dispositivo. El último argumento indica la localización del sistema de archivos en la parte cliente. Hemos compartido exactamente el `/usr/local` de *charly* con *jill* y así podemos evitar el tener que instalar programas en `/usr/local` más de una vez. Para hacer esta configuración permanente, podemos especificarlo en el archivo `/etc/fstab` de *jill*. `fstab` contiene todos los dispositivos que serán montados en el arranque. La sintaxis para `/etc/fstab` es:

```
#device mount point file system options dump fsckorder
charly:/usr/local /usr/local nfs nosuid,hard,intr 0 0
```

Sin embargo, deberá tener cuidado con una entrada permanente. Sólo podrá usarlo cuando el servidor (*charly*) esté siempre encendido, o cuando encienda *charly* antes que *jill*.

## PRECAUCIONES

Uno de los mayores problemas con NFS viene del hecho de que exista por defecto una relación de confianza entre un cliente y un servidor NFS. En el caso de que la cuenta *root* del servidor esté

comprometida, la del cliente también lo estará. El NFS-COMO describe un conjunto de medidas esenciales que debe tomarse para conseguir cierta seguridad..

Un cliente no debe confiar ciegamente en un servidor, por ello debemos especificar opciones restrictivas cuando usamos el comando mount. Ya hemos mencionado la primera de ellas: nosuid. Cancela el efecto de los bits SUID y GID. Con esto alguien que esté como *root* en el servidor primero debe hacer login en el cliente como un usuario normal y después hacerse *root*. Otra opción, más restrictiva, es noexec. Prohíbe ejecutar programas en sistema de archivos exportado. Esta opción únicamente se utiliza en sistemas que sólo contengan datos.

En el lado del servidor NFS, podemos especificar que no confíe en la cuenta *root* del cliente. Tenemos que especificarlo en `/etc/exports` con la opción `root_squash`. Entonces si un usuario con UID 0 (*root*) en el cliente accediese al sistema de archivos exportado por el servidor, tomaría el UID *nobody* para consultar archivos. Esta opción está activada por defecto bajo Linux pero se puede desactivar con la opción `no_root_squash`. Se puede especificar qué opciones deben aplicarse en un conjunto de UIDs. Recuerde también que las opciones `anonuid` y `anongid` permiten cambiar los UID/GID de *nobody* por los de otro usuario diferente.

Algunas opciones son más generales y se efectúan por el portmapper. Por ejemplo, prohibimos el acceso a todas las máquinas con la siguiente línea en el archivo `/etc/hosts.deny`:

```
# hosts.deny : absolute prohibition for every one to use the portmap
portmap: ALL
```

Después en el archivo `/etc/hosts.allow` esta estricta prohibición se puede contrarrestar permitiendo el acceso a las máquinas deseadas. Unas buenas reglas de cortafuegos también contribuyen a una protección mejor. Observe los puertos usados por los diferentes servicios y los protocolos utilizados:

SERVICIO RPC	PUERTO	PROTOCOLOS
portmap	111	udp / tcp
nfsd	2049	udp
mountd	variable	udp / tcp

## USANDO NIS, NFS Y AUTOFS

Fijémonos ahora en una red algo más complicada, como la que podrá encontrar e.g en una empresa. En una pequeña red en su casa probablemente pueda vivir sin NIS. El Servicio de Información en Red, NIS (Network Information Service) es una forma de distribuir archivos de configuración (e.g en /etc) a otras máquinas.

El servidor principal en nuestra red se llamará "*chicas*" y otras 3 máquinas de la red serán "*luisa*", "*laura*" y "*karla*". Configuramos *charly* como un servidor NIS para el dominio *lolas*. Las otras máquinas son tan sólo clientes NIS de *chicas* (podríamos tener un servidor NIS secundario, pero hoy no es ése nuestro propósito).

Primero, veamos la configuración de nuestro servidor *chicas*. Empezamos definiendo algunos mapas NIS que contienen toda la información necesaria. El archivo `/etc/netgroup` contiene los grupos de máquinas con características comunes (una misma arquitectura, por ejemplo). Un mapa NIS es muy útil para NFS. Sólo tenemos que reunir todas las máquinas permitiéndoles acceder al mismo sistema de archivos exportado. Entonces se usa este grupo en el archivo `/etc/exports` en lugar de especificar todos los clientes uno a uno:

```
# /etc/netgroup chicas (luisa,,) (laura,,) (karla)
```

Por lo que concierne a NFS, sabemos que la configuración es bastante restrictiva. El archivo `/etc/exports` de *chicas* contiene:

```
# /etc/exports /usr/local @chicas(ro)
```

Decidimos utilizar automount para acceder al directorio exportado `/usr/local`. En vez de montar el sistema en el arranque, se hace automáticamente cuando un usuario accede a un archivo de este directorio. Creamos el archivo `/etc/auto.map` para decidir qué será accesible por automount y NIS: `# /etc/auto.map charly charly:/usr/local`

Como queremos integrar esta información (el nuevo `auto.map` y los archivos `netgroup`) en la base de datos NIS, tenemos que modificar el `Makefile` antes de reconstruirlo. Debemos estar seguros de qué grupo accederá a la base. Por lo que respecta a `auto.map`, este archivo no está definido por defecto. Sólo tenemos que añadir una entrada en el `Makefile`, con la regla asociada (usando la existente como un modelo):

```
#To be added in the Yellow Pages Makefile

AUTO_MAP = $(YPSRCDIR)/auto.map
# ...
#...

auto.map: $(AUTO_MAP) $(YPDIR)/Makefile

@echo "Updating $@"
-@sed -e "/^#/d" -e s/#.*$$// $(AUTO_MAP) | $(DBLOAD) \
-i $(AUTO_MAP) -o $(YPMAPDIR)/$@ - $@
-@$(NOPUSH) || $(YPPUSH) -d $(DOMAIN) $@
```

Esta regla borra comentarios, añade una nueva entrada a la base de datos y después transmite la información a cada servidor, sólo tenemos que ejecutar make desde el directorio /var/yp.

Ahora, tenemos tres clientes: *luisa*, *laura* y *karla*. Tenemos que decir a autofs que controle un nuevo mapa dado por YPs. En el archivo /etc/auto.master de cada cliente la siguiente línea informa de la presencia de una asignación auto.map obtenida vía los servicios de YP.

```
#/etc/auto.master

/usr/local yp auto.map --intr,nosuid,nodev
```

Después tenemos que reiniciar autofs para hacer este nuevo mapa efectivo. Ahora tenemos un único directorio físico /usr/local en *chicas*. Entonces, cuando se instale un programa específico en *chicas*, todas las máquinas lo podrán usar.

Este ejemplo podría llevarse más lejos con la instalación de un único sistema /usr, /usr/doc u otros, pero la práctica muestra que no sería buena idea. Las instalaciones a menudo necesitan modificar archivos en el directorio /etc o en otros. Tendríamos que actualizar todas las máquinas para actualizar archivos no exportados, lo que sería tremendamente pesado.

## UNIDAD XII - SERVICIO SAMBA

### INTRODUCCIÓN

Samba permite compartir entre máquinas windows y linux recursos . Siendo un recurso una carpeta o la impresora. Su demonio es `smbd` permite que las máquinas windows puedan acceder a linux. Para ello el servidor deberá indicar qué carpetas quiere compartir con windows.

### INSTALACIÓN DEL SERVIDOR SAMBA

Usando `apt`, con el comando: `# apt-get install samba`

Durante la instalación nos pedirá el nombre del grupo de trabajo que llamaremos CEP (por ejemplo). El resto de opciones las dejamos por defecto.

### CONFIGURACIÓN DEL SERVIDOR

Para ver el archivo de configuración: `# cat /etc/samba/smb.conf`

Para editar el archivo de configuración: `# gedit /etc/samba/smb.conf`

Hay que cambiar el archivo de configuración de samba para que se adapte a nuestra red. Este archivo deberá grabarse en `/etc/samba`. Crearemos una carpeta que será pública, es decir, que la verá cualquier máquina, llamada `/compartido`: `mkdir /compartido`.

Cambiamos los permisos de acceso al directorio con: `chmod 777 /compartido` que concede permisos de lectura, escritura y ejecución a todo el mundo.

El archivo donde se guarda la configuración de Samba es `/etc/samba/smb.conf`.

<u>OPCIONES</u>	<u>DESCRIPCIÓN</u>
guest ok	Define si se permitirá el acceso como usuario invitado. El valor puede ser Yes o No.
public	Es equivalente del parámetro guest ok, es decir, define si se

	permitirá el acceso como usuario invitado. El valor puede ser Yes o No.
browseable	Define si se permitirá mostrar este recurso en las listas de recursos compartidos. El valor puede ser Yes o No
writable	Define si se permitirá la escritura. El valor puede ser Yes o No.
read only.	Define que sólo se tendrá permiso de lectura. El valor puede ser Yes o No.
valid users	Define qué usuarios o grupos pueden acceder. Los valores pueden ser nombres de usuarios separados por comas o bien nombres de grupo anteceditos por una @. Ejemplo: valid users = fulano, mengano, @administradores
write list	Define qué usuarios o grupos pueden acceder con permiso de escritura. Los valores pueden ser nombres de usuarios separados por comas o bien nombres de grupo anteceditos por una @. Ejemplo: write list = fulano, mengano, @administradores
admin users	Define qué usuarios o grupos pueden acceder con permisos administrativos para el recurso. Es decir, podrán acceder hacia el recurso realizando todas las operaciones como superusuarios. Los valores pueden ser nombres de usuarios separados por comas o bien nombres de grupo anteceditos por una @. Ejemplo: admin users = fulano, mengano, @administradores
directory mask	Es lo mismo que directory mode. Define qué permisos tendrán los subdirectorios creados dentro del recurso. Ejemplos: directory mask = 1777
create mask	Define qué permisos tendrán los nuevos archivos creados dentro del recurso. Ejemplo: create mask = 0644

## COMANDOS SOBRE EL SERVIDOR

Arrancando el servidor: `# /etc/init.d/samba start`

Reiniciando el servidor: `# /etc/init.d/samba restart`

¿Quién usa el servidor? `# smbstatus`

Estado del servidor: `# testparm`

Este comando comprueba el archivo de configuración del samba que es: `/etc/samba/smb.conf`

## **DANDO DE ALTA USUARIOS**

Es importante sincronizar las cuentas entre el servidor samba y las estaciones Windows. Es decir, si en una máquina con Windows ingresamos como el usuario "paco" con clave de acceso "paco", en el servidor Samba deberá existir también dicha cuenta con ese mismo nombre y la misma clave de acceso.

Como la mayoría de las cuentas de usuario que se utilizarán para acceder hacia samba no requieren acceso al interprete de mandatos del sistema, no es necesario asignar clave de acceso con el mandato `passwd` y se deberá definir `/sbin/nologin` o bien `/bin/false` como interprete de mandatos para la cuenta de usuario involucrada.

```
useradd -s /sbin/nologin usuario-windows
```

```
smbpasswd -a usuario-windows
```

Nos pedirá la clave de samba del usuario-windows. No hace falta que se asigne una clave de acceso en el sistema con el mandato `passwd` puesto que la cuenta no tendrá acceso al interprete de comandos.

Si se necesita que las cuentas se puedan utilizar para acceder hacia otros servicios como serían Telnet, SSH, etc, es decir, que se permita acceso al interprete de comandos, será necesario especificar `/bin/bash` como interprete de mandatos y además se deberá asignar una clave de acceso en el sistema con el mandato `passwd`: `# useradd -s /bin/bash usuariowindows`

```
# passwd usuariowindows
```

```
Enter new UNIX password:
```

```
Retype new UNIX password:
```

```
passwd: password updated successfully
```

```
# smbpasswd -a usuariowindows
```

Nos volverá a pedir la clave para samba del usuario-windows, que debería coincidir con la que hemos puesto como usuario linux.

### ***El archivo lmhosts***

Es necesario empezar resolviendo localmente los nombres Netbios asociándolos con las direcciones IP correspondientes. Para fines prácticos el nombre Netbios debe tener un máximo de 11 caracteres. Normalmente tomaremos como referencia el nombre corto de servidor o el nombre corto que se asignó como alias a la interfaz de red. Este lo estableceremos en el archivo `/etc/samba/lmhosts`, en donde encontraremos lo siguiente:

```
127.0.0.1 localhost
```

Debemos añadir los nombres y dirección IP del resto de las máquinas que conformen la red local. La separación de espacios se hace con un tabulador. Ejemplo: 127.0.0.1 localhost

```
192.168.1.5 maquina5
```

## **EL CLIENTE SAMBA**

Permite acceder desde Linux a recursos compartidos por máquinas windows. Para instalar el cliente samba: `# apt-get install smbclient`. Veamos qué comparte windows

```
# smbclient -L x.x.x.x
```

Siendo x.x.x.x la dirección IP de la máquina windows.

Ejemplo: `smbclient -L 192.168.100.210`, nos pedirá una clave, si la dejamos en blanco entraremos en modo anónimo.

```
Password: Anonymous login successful
```

```
Domain=[CEP] OS=[Unix] Server=[Samba 3.0.8-Debian]
```

```
.....
```



Creamos una carpeta en linux Usando

```
# mkdir /mnt/win
```

Montamos la carpeta compartida

Si la carpeta compartida por windows que se llama compartido, con linux se llamará /mnt/win

```
# mount -t smbfs //x.x.x.x/compartido /mnt/win -o username=XXX -o password=xxx
```

***Por ejemplo:***

```
# mount -t smbfs //192.168.0.210/compartido /mnt/win -o username=usuariowindows -o password=clavedeusuariowindows
```

Podemos desmontarla con: `umount /mnt/win`

***Para automatizar el proceso***

Si queremos que al encender el PC se monte automáticamente el recurso en el directorio, añadimos en el archivo **/etc/fstab** la línea `//192.168.0.210/compartidos /mnt/win smbfs`

```
auto,username=usuariowindows,password=clavedeusuariowindows 0 0
```

Recuerda que samba debe arrancar al iniciar el equipo.

***Otra forma de entrar en el entorno de red***

Podemos acceder a un PC con windows desde linux, a través de Nautilus poniendo: `smb://x.x.x.x/`

Siendo x.x.x.x la IP del PC con windows. Ponemos escribir el nombre de la máquina si le hemos asignado su IP en **/etc/hosts**.

Para entrar desde máquinas windows en recursos linux, entraríamos en windows con un login (nombre de usuario y clave) de usuario de linux, entrar en el *entorno de red*, y ver que se accede a esa máquina Linux.

Imprimir desde TRS en una impresora instalada en un pc windows. Supongamos que disponemos de una impresora **HP Laserjet 1000W** que es una Winprinter (no es compatible Linux) vamos a: instalarla en un PC con Windows XP que se llama en la red `windowsxp` y el dominio se llama "workgroup".

Desde windows, compartir la impresora con el resto de PC de la red con el nombre impresora. Configurar los PC de la red con TRS para que puedan imprimir en ella.

Veamos cómo realizar este último punto:

En el PC con TRS accedemos con el navegador a <http://localhost:631>. Entramos en "Do Administration task". Nos pedirá la clave de "root". En la página que aparece pulsamos en "Add Printer". En la nueva página pondremos el nombre que le vamos a dar a la impresora, por ejemplo:

Name: laser

Location: Conectada a windowsexp

Description: Impresora laser HP 1000W en el PC windowsexp

Pulsamos "Continue"

En la nueva página seleccionamos Windows Printer via Samba y "Continue"

En la siguiente página pondremos `smb://usuario:clave@workgroup/windowsexp/impresora` (Véase la nota) y "Continue"

En la siguiente página ponemos la marca HP y "Continue"

En la siguiente página seleccionamos el modelo HP Laserjet 4 Foomatic/ljet4 (en) y "Continue"

Si todo ha ido bien nos mostrará una página donde se nos indica que ya se ha instalado la impresora.

Para ver la lista de impresoras instaladas pulsamos en la barra superior de la página "Printers"

Comprobamos que funciona correctamente pulsando en "Print test page" y se imprimirá una página de prueba. Si entras en alguna aplicación como OpenOffice en la ventana de opciones de impresión, debería estar disponible la impresora que hemos instalado.

En `smb://usuario:clave@workgroup/windowsexp/impresora` debemos poner un usuario con su clave de windowsexp. Dado que el archivo `/etc/cups/printers.conf` contiene:

<Printer laser>

Info Impresora laser HP 1000W en el PC windowsxp

Location Conectada a windowsxp

DeviceURI smb://usuario:clave@workgroup/windowsxp/impresora

State Idle

Accepting Yes

JobSheets none none

QuotaPeriod 0

PageLimit 0

KLimit 0

</Printer>

y puede leerse por todos los usuarios, plantea un problema de seguridad. Para evitar el problema podemos: Impedir la lectura de este archivo usando chmod. Usar el usuario "invitado" sin clave poniendo [smb://invitado:@workgroup/](smb://invitado:@workgroup/windowsxp/impresora) windowsxp/impresora.

## SECCIÓN DE EJERCICIOS

### EJERCICIO N° 1

- Abre un terminal (Aplicaciones/Herramientas del Sistema/Terminal).
- Introduce los comandos explicados anteriormente y comprueba su funcionamiento.
- Verifique que el BIOS de su equipo soporta tarjetas PCI y que el arranque desde el CD es soportado.

### EJERCICIO N° 2

- Abre un terminal si cerraste el anterior.
- Usando el comando ls, comprueba el contenido de tu directorio de usuario.
- Comprueba ahora el contenido del directorio `/etc`.
- Comprueba el contenido del directorio `/`.
- Escribe `ls /etc/f*`. Observa que nos muestra los archivos y directorios cuyo nombre empieza por "f".
- Escribe el comando que muestre todos los archivos del directorio `/etc` que acaben en `".conf"`

### EJERCICIO N° 3

Haz `cat /etc/fstab`: El archivo `/etc/fstab` contiene líneas donde se indica qué dispositivo debe "montar", el lugar donde "montarlo", así como el sistema de archivos y las opciones (en este archivo, se pueden poner dos opciones más: `auto` y `noauto`, que indican si se debe "montar" automáticamente al arrancar el sistema o no, respectivamente). Más adelante puede verse su relación con el comando `mount`.

`cat -n /etc/passwd`: Contiene de claves del sistema.

`cat /etc/resolv.conf`: Contiene las direcciones DNS y el nombre del dominio.

`cat /etc/hosts` : Contiene las direcciones y los nombres de los equipos de la red.

- Busca los archivos del directorio `/etc` que contengan `".hda"`. Observa que hay archivos a los

que "grep" no puede acceder.

cat /etc/shadow: Contiene los grupos registrados en el sistema. No tenemos permiso de lectura sobre este archivo → Para poder ver su contenido podemos entrar como "root", para ello escribimos el comando su y nos pedirá la clave. Acabamos de entrar como administrador del sistema ("root"), comprueba que ahora puedes listar el contenido de "etc/shadow" con el comando "cat".

- Crea un archivo usando gedit archivo1, nos preguntará si deseamos crear el archivo y contestaremos "Si". Aparecerá la ventana del editor. Escribe dentro de él: "Curso TRS". Guarda las modificaciones pulsando en "Guardar".
- Comprueba usando **grep** que puedes encontrar ese archivo al buscar los que contengan la cadena "Curso" en el directorio "/home" y todos sus subdirectorios. ¿Lo encontraremos igual si buscamos los archivos que contengan la cadena "curso"? Escribe cat archivo1 archivo1 archivo1 > archivo2. Comprueba con **ls** que ha aparecido un archivo llamado "archivo2". → Observa usando **cat** el contenido del archivo.

#### EJERCICIO N° 4

Observa el resultado de escribir los comandos ls -la /dev. Como puedes ver el listado es demasiado extenso y aunque movamos la barra de desplazamiento vertical de la ventana del terminal, no podemos llegar al principio del listado. Observa el resultado de ls -la /dev | more. Comprobarás que es más fácil ver el listado de archivos.

#### EJERCICIO N° 5

- Escribe el comando cp -i /etc/resolv.conf ejercicio5.
- ¿Cuántos Mbytes tiene en uso su directorio home?
- Escribe el comando **ls** y comprueba que se ha copiado el archivo "resolv.conf" a tu directorio de trabajo con el nombre "ejercicio5".
- Copia los archivos /etc/passwd /etc/fstab a tu directorio de trabajo.
- Cambia sus nombres a "claves" y "tabla", respectivamente.
- Borra los tres archivos: "claves", "tabla" y "ejercicio5".

- Copia el directorio `/etc/apt` a tu directorio de trabajo con el nombre `apt_copia`.
- Comprueba con `ls /etc/apt` y `ls apt_copia`, que el contenido de los dos directorios coincide.
- Cambia el nombre del directorio `apt_copia` a `apt_copia_2`.

## EJERCICIO Nº 6

- Crea en tu directorio de trabajo los subdirectorios: `carpeta1`, `carpeta2` y `carpeta3`.
- Usando `cd` colócate en el directorio `carpeta1` y crea dentro de él los subdirectorios: `carpeta11` y `carpeta12`.
- Crea los subdirectorios: `carpeta2/carpeta21` (es decir, crea dentro del directorio `carpeta2` el subdirectorio `carpeta21`) y `carpeta2/carpeta22`.
- Crea los subdirectorios: `carpeta3/carpeta31`, `carpeta3/carpeta32` y `carpeta3/carpeta33`.
- Borra el subdirectorio `carpeta2/carpeta21`.
- Cambia el nombre del subdirectorio `carpeta22` a `carpeta222`.
- Copia el directorio `apt_copia_2` dentro de la carpeta `carpeta3`.
- Empaqueta el directorio `carpeta_3` y todo su contenido (incluidos los subdirectorios) en el archivo `comprimido.tar`.
- Borra todos los directorios que has creado.
- Comprueba con `ls` que han desaparecido los directorios que hemos borrado.
- Desempaqueta el archivo `comprimido.tar` en tu directorio de trabajo.
- Comprueba que ha vuelto a parecer el directorio `carpeta3` con todo su contenido (incluidos subdirectorios).
- Vuelve a borrar el directorio `carpeta3`.
- Comprime el archivo `comprimido.tar` con `gzip -9 comprimido.tar`. Haz `ls` y comprueba lo que ha ocurrido con el archivo `.tar`.
- Descomprímelo usando `gunzip`. Comprueba con `ls` que ha vuelto a aparecer el archivo `.tar`.

- Desempaqueta ese archivo y comprueba que ha vuelto a aparecer el directorio "carpeta3".

## EJERCICIO N° 7

- Haz un enlace débil de "/home/usuario/enlace" a "/home/usuario/carpeta3".
- Haz ls -la enlace y ls -la carpeta3. Comprueba que su contenido es el mismo.
- Haz ls -l y comprueba que en el lista aparece **enlace -> /home/usuario/carpeta3**.
- Crea un directorio llamado "lil" dentro del directorio "enlace".
- Comprueba que en el directorio "carpeta3" también aparece ahora ese directorio, es decir, lo que hagamos en una carpeta se verá reflejado en la otra, y viceversa.
- Crea con OpenOffice un documento SXW dentro de tu directorio de trabajo y escribe cualquier contenido.
- Crea en /tmp un enlace fuerte a dicho archivo y comprueba lo que ocurre cuando modificas dicho archivo.

## EJERCICIO N° 8

- Escribe who y comprueba los usuarios que se encuentran activos en el sistema.
- Escribe finger usuario y comprueba la información que puedes obtener del usuario.
- Entra como superusuario con "su" y crea el usuario "pepe" con clave "pepe".
- Termina tu sesión pulsando en "Acciones > Terminar sesión" y accede como usuario "pepe" con su clave. *Cuando termines, pulsa cerrar la sesión de "pepe" y vuelve a "usuario"*.
- Crea otro usuario llamado "jose" con clave "jose".
- Borra el usuario "jose".
- Comprueba que el directorio "/home/jose" sigue existiendo. Para liberar espacio en disco borra ese directorio.
- Crea el grupo **alumnos**.
- Añade el usuario "pepe" al grupo **alumnos** con adduser pepe alumnos. Para poder ver los

grupos y comprobar qué usuarios pertenecen a él podemos ver el contenido del archivo **/etc/group**

- Elimina a "pepe" del grupo "alumnos" con `deluser pepe alumnos`.
- Comprueba de nuevo los usuarios que pertenecen al grupo "alumnos".
- Elimina el grupo "alumnos".

## EJERCICIO Nº 9

- Abre un terminal.
- Escribe **gedit archivo** para crear el archivo "archivo" y guardarlo en tu directorio de trabajo. Cierra la ventana del editor.
- Comprueba los permisos del archivo `ls -l archivo`, que serán **-rw-r--r--**.
- Cambia los permisos de "archivo" a: **"-rw-rw-r--"**
- Cambia los permisos de "archivo" a: **"-rw-r--rw-"**
- Cambia los permisos de "archivo" a: **"-rwxrwxrwx"**
- Cambia el propietario de "archivo" a "pepe"
- Cambia el grupo de "archivo" a "pepe"
- Muevete al directorio **/home** y comprueba usando `ls -l` los permisos del directorio "pepe" (que es el directorio de trabajo del usuario "pepe"). Verás que se obtiene algo como: `drwxr-xr-x 30 pepe pepe 4096 2005-01-18 10:15 pepe`. Por defecto estos son los permisos que el sistema pone a los directorios de trabajo de los usuarios que se crean en él.
- Comprueba que puedes acceder al contenido del directorio del usuario "pepe" con `ls -la /home/pepe`. Para poder modificar los permisos del directorio `"/home/pepe"` vamos a entrar como superusuario con **su**.
- Ahora debemos cambiar los permisos del directorio `"/home/pepe"` a `drwx-----`. Dejamos de ser superusuario con `exit`.
- Comprueba que ahora como usuario "usuario" ya no puedes acceder al directorio `"/home/pepe"`.



- Comprueba que puedes cambiar los permisos de tu directorio de trabajo `"/home/usuario"`.
- Vuelve a dejar los permisos de `"/home/usuario"` como estaba, es decir, `drwx-----`. Sería recomendable informar a cada usuario que creemos de esta situación, para que modifique los permisos de acceso a su directorio de trabajo, si así lo cree conveniente. O que modifiquemos como superusuario dichos permisos.

## EJERCICIO N° 10

- Con la orden que corresponda comprueba la lista de repositorios de tu PC.
- Usando el comando que corresponda, comprueba si existe algún paquete con el nombre parecido a `"fortune"`.
- Abrimos un terminal e iniciamos sesión como root con el comando **`su`**.
- Ejecuta en ese terminal la orden que actualice la lista de paquetes de tu PC `apt-get update`.
- Busca todos los paquetes que contienen el nombre `"fortune"`. Observa que en la lista que aparece hay uno llamado `"fortunes-es"`.
- Instala el paquete `"fortunes-es"`.
- Abre otro terminal y escribe **`fortune`**. Repite este comando varias veces.
- Comprobarás que muchas frases son "políticamente incorrectas". Así que desinstala `"fortune-es"`.
- Cierra todas las terminales.

## EJERCICIO N° 11

Repite el ejercicio N° 10 pero utilizando `aptitud`.

## EJERCICIO N° 12 - *Compartiendo un directorio (en toda la red local).*

- Crea un directorio llamado `Compartido_nfs` (o el nombre que prefieras) dentro de tu directorio de trabajo `mkdir Compartido_nfs` (o `mkdir /home/usuario/Compartido_nfs`).
- Inicia sesión de root con el comando `su` y edita (con `nano`, `vi`, `gedit`, etc.) el archivo `/etc/exports`, para añadir una línea como la siguiente: `/home/usuario/Compartido_nfs *(ro,sync)`.

- Reiniciamos el *servidor nfs* `/etc/init.d/nfs-kernel-server restart`

### **EJERCICIO N° 13 - *Compartiendo un directorio con permiso de lectura y escritura***

- Creamos ahora, como usuario, sin iniciar sesión de root el directorio `/home/usuario/LE` (LE abreviación de *lectura y escritura*).
- ¿Qué tenemos que escribir en `/etc/exports` para compartirlo en modo `rw`?
- ¿Qué comando tenemos que ejecutar, tras editar este archivo, para activar los cambios en el servidor NFS?
- Crea algún directorio y monta en él el directorio `/home/usuario/LE` compartido por alguno de tus compañeros. ¿Puedes crear alguna carpeta en este directorio? ¿En qué PC se crea dicha carpeta?

### **EJERCICIO N° 14 - *Opciones de inicio en /etc/fstab***

- Edita el archivo `/etc/fstab` y establece las siguientes opciones para el arranque: El directorio remoto `192.168.0.250:/mnt/hda5/Compartido_NFS` se montará automáticamente en `/mnt/nfs`. El directorio remoto `192.168.0.250:/home/usuario/Pruebas` podrá ser montado por cualquier usuario, en modo `rw` en el directorio (que habrá que crear) `/home/usuario/Prueba_remota`

### **EJERCICIO N° 15 - *Accediendo con permiso de root (muy peligroso)***

Vamos a compartir el directorio `/home` de forma que el usuario root (`uid=0`) remoto tenga sobre él permisos de administrador. Esto es muy peligro pues si alguien conecta a la red local otra máquina en la que es capaz de iniciar sesión como root, podría acceder a nuestro `/home` con total impunidad. Para ello:

- Edita el archivo `/etc/exports` y escribe la siguiente línea `/home *(rw,no_root_squash,sync)`.
- Reinicia el servicio NFS (`/etc/init.d/nfs-kernel-server`).
- Crea en una máquina remota (o en el propio equipo si sólo quieres hacer pruebas) el directorio `/mnt/prueba_home`.
- Monta en `/mnt/prueba_home` el directorio `/home` compartido mediante NFS: `mount -t nfs 192.168.0.74:/home /mnt/prueba_home` (pon la IP que corresponda en tu caso).

- Accede, como root a /mnt/prueba\_home y prueba a crear directorios, renombrar archivos, entre otros. ¡***Mucho cuidado!***