INDICE

SHELL

Todos los intérpretes de comandos de Unix (*shell*), se han portado a **GNU/Linux**. Un intérprete de comandos es la *interface* de comunicación entre el usuario y el Sistema Operativo. Existen numerosos *shell*: **csh** (C Shell), **sh** (Bourne Shell), pero, quizás, el más extendido entre los usuarios e implementaciones de Linux es el **bash** (Bourne Again Shell). La principal diferencia entre los *shell* estriba en las funciones que aportan, instrucciones, funcionalidad en general. Y esta es la razón dominante que determinará que intérprete de comandos utilizar.

Si comparamos este concepto en Linux con el Sistema Operativo MS-DOS, el intérprete de comandos sería, en este, ***COMMAND.COM***. Aquí ***command.com*** aporta una serie de funciones internas (por ejemplo, ***dir***) y funciones o programas externos que se ejecutan a través del intérprete. Tanto ***csh***, ***ksh***, ***sh***, ***bash*** o cualquiera de los intérpretes tienen esa misma función. Aportan funciones/órdenes internas, son la *capa* que, sobre el Sistema Operativo, permite la invocación del resto de órdenes del sistema y de programas externos.

Además, los intérpretes de Linux tienen funciones que permiten escribir secuencias de programación (*shell scripts*) de una gran sofisticación que simplifican la automatización de muchas tareas sin necesidad de escribir un programa en otro lenguaje específico.

INTÉRPRETES DE COMANDOS

Un intérprete de comandos es un programa (que reside en /bin), que lee las entradas del usuario (por ejemplo las órdenes que teclea) y las traduce a instrucciones que el sistema es capaz de entender y utilizar.

> El intérprete de comandos no es el único interface para Linux. Existen otras interfaces posibles, como el sistema X Windows, el cual le permite ejecutar comandos usando el ratón y el teclado.

Tan pronto como entra en el sistema, éste arranca un interprete de comandos, momento a partir del cual ya puede teclear órdenes al sistema.

SINTAXIS DE UNA ORDEN

orden ***-opciones*** **argumentos**

La orden va en minúsculas, las opciones suelen ir precedidas por un guión (-) y pueden ser más de una; los argumentos pueden ser varios separados por espacios. Algunas órdenes no admiten opciones, otras precisan dos tipos de argumentos (origen y destino), otras no admiten ni opciones ni argumentos. Ver *figura 1*.

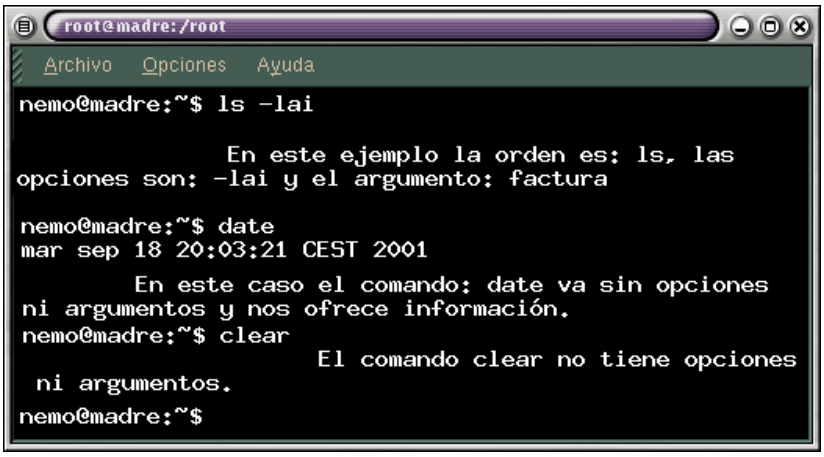


figura 1

Si tecleamos una orden y el intérprete de comandos no puede encontrar el programa de ese nombre dado en la orden, se muestra un mensaje de error que debería de ser autoexplicativo, como se puede ver en la *figura 2* al intentar ejecutar la orden "explorer"

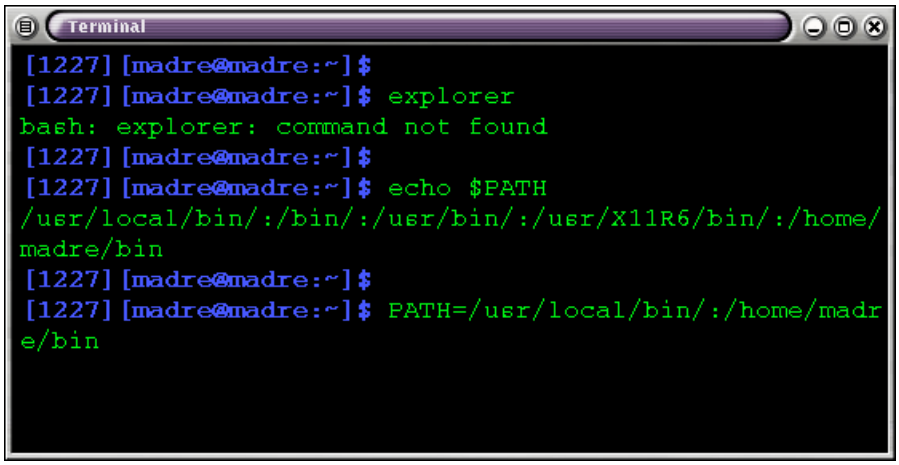


figura 2

A menudo verá este mensaje de error si se equivoca al teclear una orden o si el directorio donde está el programa a ejecutar no está declarado en el PATH.

PATH Es una variable del sistema. En el PATH se indican las rutas, es decir, los directorios donde el interprete de comandos debe buscar las órdenes del sistema. En la *figura 5* se muestra la forma de visualizar la variable PATH con el comando "echo" y también una forma sencilla y volátil de declararla de nuevo (al menos para esa shell.. variar el valor de variables como PATH puede tener efectos inesperados y poco deseados)

Para cada usuario se inicia una copia del shell en cada sesión. Dos consolas virtuales significan dos copias del shell independientes entre sí, aunque sean del mismo usuario. Además para cada orden que se ejecute el shell genera un sub-shell que «muere» al finalizar la ejecución de la orden. También se pueden lanzar nuevos shells (o subshells) dentro de una shell con la orden "**sh**" (o con el nombre de la shell en cuestión que queramos lanzar, por ejemplo "**bash**").

METACARACTERES

Estos, llamados comodines, le permiten referirse a, por ejemplo, grupos de ficheros que tienen algún tipo de coincidencia.

El comodín " * " (asterisco) hace referencia cualquier carácter o cadena de caracteres.

El uso de este comodín **NO** cuadrara con nombres de ficheros que comiencen con un punto («**.**»). Estos ficheros son tratados como «ocultos» aunque no están realmente ocultos, simplemente no son mostrados en un listado normal de ls y no son afectados por el uso del comodín *****. Si «*****» coincidiera con ficheros que comienzan por «.» actuaría sobre «**.**» y «**..**». Esto puede ser peligroso con ciertas ordenes, (como por ejemplo: rm *****).

El comodín " **?"** **este carácter solo expande un único carácter.**
El comodín [**..**] s**ustituye cualquier valor incluido entre los corchetes.**

BASH (FUNCIONES AVANZADAS)

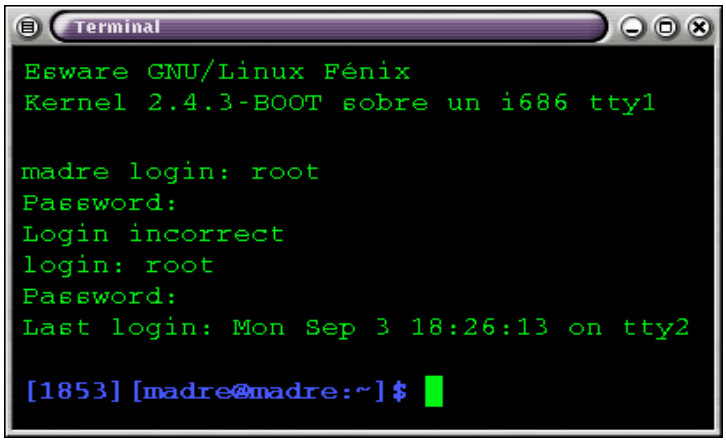
Se denomina **shell** (significa concha en inglés) al interprete de comandos. En realidad es uno de los programas más importantes, grandes y potentes que ejecuta cualquier máquina **Linux** desde que se levanta el sistema operativo. Tanto si lo usamos como "interface permanente para la introducción de órdenes", como si usamos una interface gráfica (de igual modo **bash** estará corriendo "debajo").

Hablaremos preferentemente de **bash** (Bourne Again Shell), porque es la que utilizan todas las distribuciones de GNU/Linux y algunos UNIX comerciales muy conocidos.

Por tanto, **bash** será nuestro intérprete y traducirá al **kernel** (núcleo) de ESware Linux aquellas órdenes que nosotros vayamos pasándole a través del teclado o el puntero del ratón. Pero la **shell** hace algunas cosas más que interpretar comandos. Las veremos después de saber como se introduce una orden.

EL PROMPT Y LA SINTAXIS

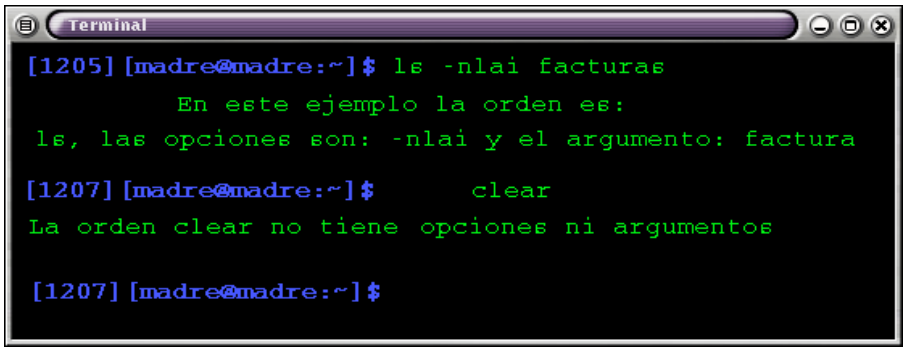
Lo primero que vamos a ver en la pantalla de nuestro monitor (siempre que no tengamos predefinido un arranque gráfico con **xdm**, **gdm** o **kdm** y la máquina haya arrancado correctamente), será la **shell bash**, ejecutando el programa: **/sbin/mingetty** en la primera consola (**tty1**). Es el programa que se encarga de identificarnos ante el sistema (login). Nos pide un nombre de usuario y una contraseña válida, es decir, se asegura de que tengamos una cuenta activa (y no en el banco precisamente). Si introducimos estos dos datos y **bash** comprueba que son correctos, nos dará paso al sistema para que lo usemos con los recursos y privilegios que el administrador haya previsto que tengamos. Y aparecerá el **prompt** (también llamado *introductor de órdenes)*.



login

El prompt por defecto muestra el nombre del usuario, el nombre de la máquina, el directorio actual y el símbolo **$** para los usuarios y **#** para el administrador. El prompt puede cambiarse modificando la variable **PS1** (ver el manual de introducción).

La sintaxis básica de las órdenes en **UNIX / Linux** es: **orden** ***-opciones*** **argumentos** . La orden debe escribirse en minúsculas (no olvidemos que Linux distingue las minúsculas de las mayúsculas), las opciones suelen ir precedidas por un guión (-) y pueden ser más de una; los argumentos pueden ser varios separados por espacios (o tabulaciones). Algunas órdenes no admiten opciones, otras precisan dos tipos de argumentos (origen y destino), y muy pocas no admiten ni opciones ni argumentos.

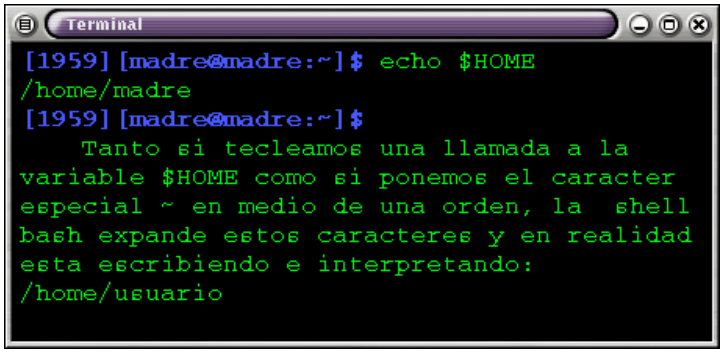


órdenes y opciones

QUE HACE BASH CON NUESTRAS ÓRDENES

Lo cierto es que no pasa al núcleo la orden tal y como nosotros se la escribimos. Digamos que las "interpreta" además de traducirlas a un lenguaje comprensible para el **kernel**.

Primero la expande. Es decir, si nosotros tecleamos alguno de los caracteres que tienen un significado especial o una ***variable***.

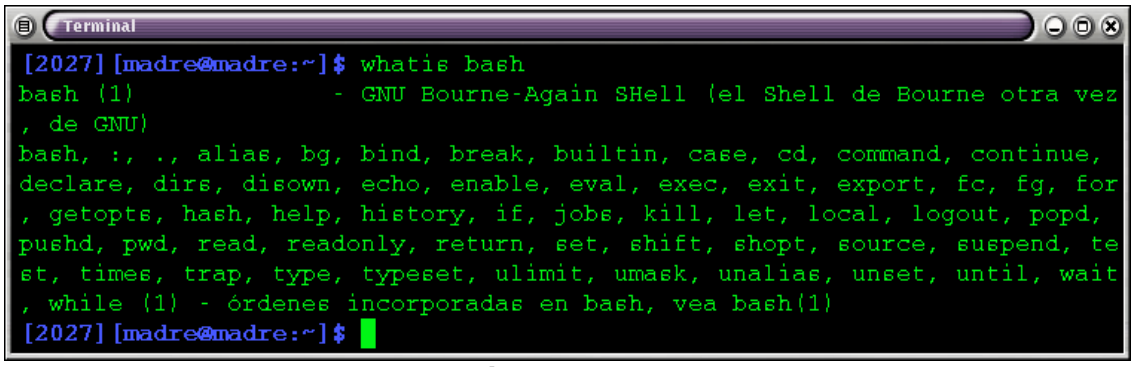


home

COMANDOS INTERNOS

A continuación **bash** comprueba si la orden (o las órdenes, porque pueden ser varias seguidas o entubadas), son internas o externas.

Las órdenes internas son las que el intérprete de comandos sabe ejecutar por si mismo. Si queremos saber los comandos internos de **bash**, debemos de teclear alguno de los comandos informativos, por ejemplo, **whatis** o apropos . Algunos de los comandos que figuran como internos en realidad son "internos y externos", es decir, además de figurar entre los que la shell puede interpretar, también existe el ejecutable en alguno de los directorios a tal efecto.



comandos internos

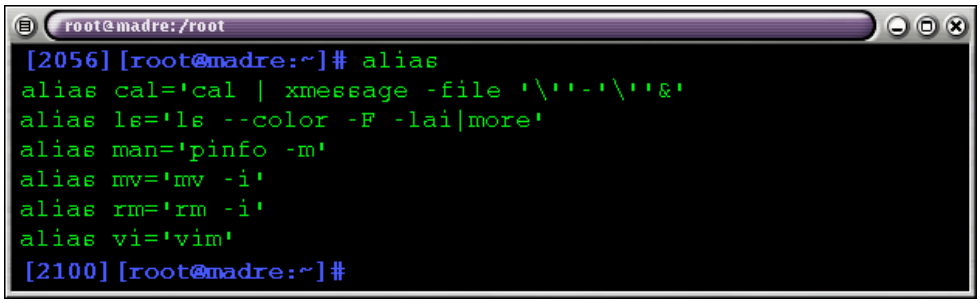
Algunas de las órdenes que aparecen haciendo un: **whatis bash** como **kill** o **echo** son comandos que además figuran como ejecutables en el directorio **/bin**. De tal forma que, si los renombramos se ejecutarán los internos de la shell. Este comportamiento no es gratuito y los que programan scripts en shell deben de tener muy en cuenta esta circunstancia.

LOS ALIAS

bash comprueba además si la orden que hemos escrito es un "**alias**" (un nombre que en realidad está sustituyendo a otro comando, a más de uno o a un comando con opciones. El comando **alias** se utiliza para definir "sobrenombres" para otras órdenes y comandos. Si queremos saber los alias definidos para nuestro usuario, simplemente tecleamos el comando sin argumentos (se trata de comando interno de la shell). Si queremos definir un alias lo hacemos escribiendo el nombre real y el "sobrenombre" después del comando. Estos alias se definen en los archivos "de perfil" de cada usuario **/home/user/bashrc** o **/home/user/bash_profile**, mejor en el primero y como se ve en el ejemplo. De esta forma al teclear la orden **rm** la shell traducirá y ejecutará **rm -i**, así el comando **rm** pedirá confirmación antes de borrar. Para deshacer un alias se usa el comando **unalias**, con esta sintaxis :

unalias [alias que se definió con "alias"] .

Si habíamos definido: **alias vi='vim'**, lo desharíamos con: **unalias vi.**



alias

EL PATH

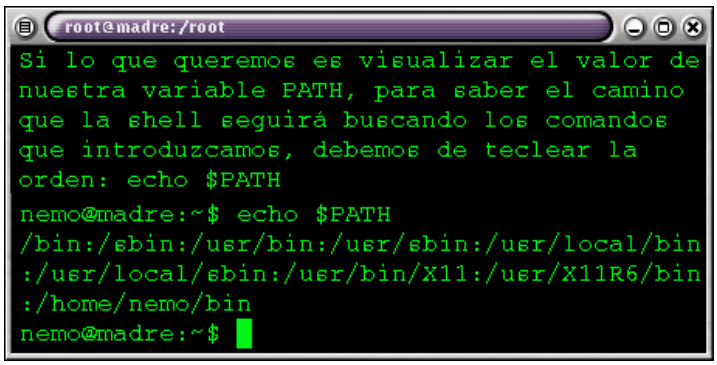
Si no se trata de un comando interno el interprete busca el programa (o binario) en el **PATH** y lo ejecuta pasándole las opciones y argumentos especificados en la línea de comandos.

En el caso de que **bash** no encontrase la orden, es decir, que no se cumpla ninguna de las condiciones que hemos especificado anteriormente (no es un comando interno, no es un alias y no está en el path), nos mostrará un mensaje de error diciéndonos que el comando no ha sido encontrado.

Debemos detenernos un momento para explicar que es el **PATH** (camino o ruta en inglés). Todos los comandos que no son internos de la shell, son binarios ejecutables que se encuentran en directorios determinados. Algunos de esos directorios contienen comandos que pueden ejecutar todos los usuarios del sistema y otros contienen binarios que sólo el administrador (root) puede usar.

En los archivos de "perfil" que residen en el /home/user de cada usuario se define **PATH**, como una variable y es esa variable la que contiene los directorios con los comandos que ése usuario en particular puede usar. Es decir, aquellos directorios en los que **bash** buscará los comandos que teclee ese usuario. Si los encuentra los ejecutará, si la orden introducida no está en el path del usuario la shell mostrará un mensaje de error.

El administrador puede redefinir las variables **PATH** de cada usuario. A título informativo diremos que para ver todas las variables declaradas, podemos teclear los comandos: **set** y env .



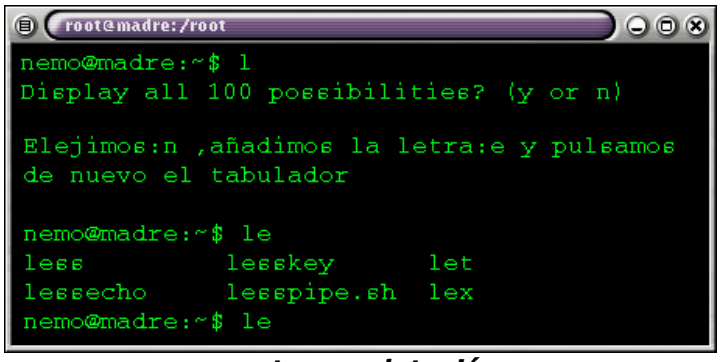
variables

AUTOCOPLETACIÓN

En el caso de que no hayamos tecleado una orden correctamente, también aparecerá el mismo mensaje de error (es una de las causas más comunes de este tipo de mensajes). El intérprete de comandos nos ofrece otra herramienta denominada "autocompletion", que podemos utilizar para completar las órdenes que vayamos introduciendo en la línea de comandos.

Nosotros escribimos la primera parte de la palabra y pulsando el tabulador la shell la completará, **si existe**. Funcionará tanto si lo que tecleamos es una orden como los argumentos de la misma. Si al teclear una ruta de directorios como argumento de un comando, el tabulador no completa el nombre del directorio o del fichero, podemos estar seguros de que, o no existe o no está en esa ubicación.

Puede darse el caso de que el tabulador no complete la palabra en la primera pulsación. Si pulsamos de nuevo el tabulador veremos el motivo, seguramente hay más de un archivo o directorio que contienen esa parte de la palabra. En ese caso, a la segunda pulsación la shell nos muestra todas las coincidencias para que escojamos la correcta. Si hay demasiadas coincidencias, nos preguntará si queremos que nos muestre las, por ejemplo, 250 posibilidades y nosotros debemos decidir.



autocompletación

LA HISTORIA DE LOS COMANDOS

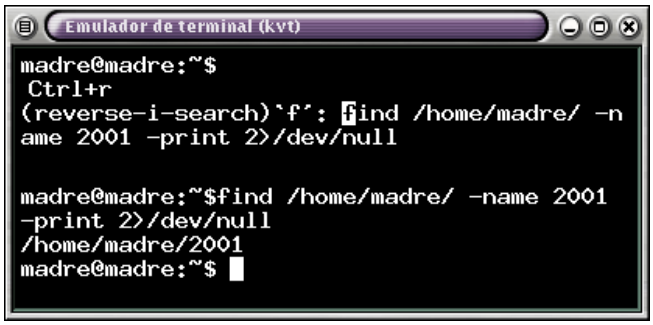
Todavía no hemos terminado con todo lo que **bash** tiene que hacer cada vez que le damos una orden. A veces resulta difícil asumir que la shell haga tantas cosas y tan rápido, pero hay que tener en cuenta que es uno de los programas más importantes que se ejecutan en nuestra máquina desde que arrancamos el sistema.

Cada vez que introducimos una orden, **bash** la escribe en un archivo que reside en el /home/usuario y se llama **.bash_history**. ESware Linux por defecto determina que se guarden en ese fichero los últimos 1000 comandos utilizados. De manera que se van añadiendo los más recientes y borrando los más antiguos para que siempre estén disponibles las 1000.

Podemos leer este archivo histórico de órdenes simplemente editándolo con **vim**, pero **bash** nos ofrece formas más rápidas de acceso y completación de comandos. Una de ellas es a través de las teclas de dirección de nuestro teclado. Pulsando la tecla que mueve el cursor hacia arriba la shell nos sitúa en el prompt las últimas órdenes, desde las más recientes a las más antiguas. Pulsando la tecla que mueve el cursor hacia abajo, navegamos por el histórico de comandos en dirección contraria.

CTRL + R

Otra forma de acceder al histórico de órdenes es pulsando la combinación: **Ctrl+r** . Desaparece el prompt y en su lugar muestra la frase: **(reverse-i-search)`':** por cada letra que vayamos introduciendo la shell nos irá mostrando aquellas órdenes que contengan esa cadena de caracteres, sólo tenemos que pulsar **Intro** para que se ejecuten de nuevo.



Ctrl+r

ENTRADAS, SALIDAS, REDIRECCIONES Y TUBERÍAS

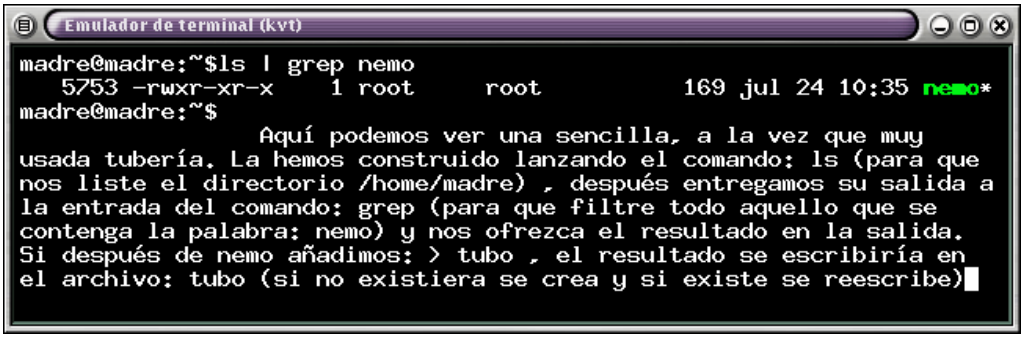
La shell nos proporciona la posibilidad de ejecutar comandos "seguidos", es decir, uno a continuación de otro. Esto lo conseguimos escribiendo los comandos seguidos y separados por el operador " **;** " .

varios comandos

También nos ofrece la facilidad de "entubar" comandos, es decir , ejecutando varios comandos seguidos pero, entregando la salida de uno a la entrada de otro. De manera que la ejecución del segundo queda condicionada a la salida del primero y así sucesivamente. Para lanzar comandos de esta manera debemos usar el operador " **|** " (**AtlGr+1**), o "pipe" entre ellos. Se pueden hacer tuberías con la mayoría de los comandos.

Además de poder redirigir la salida de un comando hacia la entrada de otro, **bash** también nos permite redirigir las salidas a ficheros. Para poder hacer uso de esta posibilidad tenemos los operadores: " **<** " que indica entrada standard (la entrada standard es el teclado si no indicamos otra ..) , " **>** " que indica salida standard (es lo mismo que **1>** y se refiere a la consola, si no indicamos otra cosa) y " **2>** " que se refiere

a la salida standard de errores (se refiere a la consola a no ser que indiquemos otro destino). Los operadores de redirección son muy potentes y hay que manejarlos cuidadosamente. Si redirigimos la salida de un comando a un archivo y este archivo existe, su contenido será machacado.



tuberías (pipes)

Podemos decir que para hacer todas estas operaciones, la shell escribe las salidas de los comandos en algo parecido a un "fichero virtual" y si tiene que redirigir esta salida a la entrada de otro comando, hace que ése comando tome la entrada de ese "fichero virtual" en lugar de tomarlo de la entrada standard (normalmente el teclado). De esa forma puede hacer las redirecciones y las tuberías.

OPERADORES ESPECIALES

La shell nos proporciona además otros mandatos especiales (sin entrar a comentar los utilizados normalmente en la programación de scripts . Uno de ellos es el " **.** " , si lo introducimos inmediatamente antes del comando o ejecutable, **bash** entiende que debe ejecutar todas las órdenes del archivo que se le da como argumento de del comando.

Si escribimos: **. /** delante del comando o ejecutable la shell entiende que debe de buscar ese ejecutable en el directorio actual, es decir, independientemente de que en el **PATH** o en los comandos internos de la propia shell haya un ejecutable con el mismo nombre, **bash** ejecutará el que esté en el directorio de trabajo.

Otro de los operadores especiales que se usan habitualmente es: " **&** ". Si lo introducimos a continuación de un comando, la shell lo ejecutará en segundo plano. De esta manera podemos seguir usando la shell, porque nos devuelve el prompt. Evidentemente no debemos ejecutar en segundo plano comandos interactivos (como **BC**), porque no veríamos el segundo prompt (**PS2**), que la shell devuelve para que le pasemos los argumentos.

ARQUITECTURA DE UN ORDENADOR

INTRODUCCION

Evolución de los computadores

-La era electrónica de los computadores

Los computadores envasados en elementos mecánicos planteaban ciertos problemas:

- La velocidad de trabajo estaba limitada a la inercia de las partes móviles.
- La transmisión de la información por medios mecánicos (engranajes, palancas, etcétera.) era poco fiable y difícilmente manejable.
- Los computadores electrónicos salvan estos inconvenientes ya que carecen de partes móviles y la velocidad de transmisión de la información por métodos eléctricos no es comparable a la de ningún elemento mecánico.

- Generaciones de ordenadores

1ª generación: (1946-1955) Computadores basados en válvula de vacío que se programaron en lenguaje máquina o en lenguaje ensamblados.
2ª generación: (1953-1964) Computadores de transistores. Evolucionan los modos de direccionamiento y surgen los lenguajes de alto nivel.
3ª generación: (1964-1974) Computadores basados en circuitos integrados y con la posibilidad de trabajar en tiempo compartido.
4ª generación: (1974-) Computadores Que integran toda la CPU en un solo circuito integrado (microprocesadores). Comienzan a proliferar las redes de computadores.

LA UNIDAD CENTRAL DE PROCESO

Funciones que realiza

La Unidad central de proceso o CPU, es un circuito microscópico que interpreta y ejecuta instrucciones. Se ocupa del control y el proceso de datos en los ordenadores. La CPU es un microprocesador fabricado en un chip, un único trozo de silicio que contiene millones de componentes electrónicos. La CPU se comunica a través de un conjunto de circuitos o conexiones llamado bus. El bus conecta la CPU a los dispositivos de almacenamiento (por ejemplo, un disco duro), los dispositivos de entrada (por ejemplo, un teclado o un mouse) y los dispositivos de salida (por ejemplo, un monitor o una impresora).

LA MEMORIA

Funciones que realiza

La memoria de un computador se puede definir como los circuitos que permiten almacenar y recuperar la información. En un sentido más amplio, puede referirse también a sistemas externos de almacenamiento, como las unidades de disco. Cada vez se requiere cada vez más memoria para poder utilizar los nuevos programas.

Organización

- Jerarquía de memoria

En un ordenador hay una jerarquía de memorias atendiendo al tiempo de acceso y a la capacidad. Se puede establecer la siguiente jerarquía:

- Registros de procesador: Los registros tienen un tiempo de acceso muy pequeño y una capacidad mínima (1 a 8 bytes).
- Registros intermedios: tienen un tiempo de acceso muy breve y muy poca capacidad.
- Memorias caché: Son memorias de pequeña capacidad. Normalmente una pequeña fracción de la memoria principal y pequeño tiempo de acceso.
- Memoria central o principal: Tiene un tiempo de acceso relativamente rápido y gran capacidad.
- Extensiones de memoria central: El tiempo de similar y su capacidad puede ser algunas veces mayor.
- Memorias de masas o auxiliares: suelen tener gran capacidad pero pueden llegar a tener un tiempo de acceso muy lento

- Clasificación de las memorias

Las memorias se clasifican, por la tecnología empleada y, según la forma en que se puede modificar su contenido. Las memorias se clasifican en dos grandes grupos:

1) Memorias RAM: Son memorias en las que se puede leer y escribir, (Random access memory). Por su tecnología pueden ser de ferritas o electrónicas. Para entendernos, podríamos decir que es la memoria que utiliza el ordenador para funcionar.

2) Memorias ROM: (Read 0nly Memory): Son memorias en las que sólo se puede leer. La información guardada en la ROM no pude ser cambiada por el usuario. En la ROM se guardan los bios que son los archivos necesarios para que arranque el ordenador.

BUSES DEL SISTEMA

Funciones que realiza

El bus se puede definir como un conjunto de líneas conductoras de hardware utilizadas para la transmisión de datos entre los componentes de un sistema informático. Un bus es una ruta que conecta diferentes partes del sistema, como el microprocesador, la controladora de unidad de disco, la memoria y los puertos de entrada/salida, para permitir la transmisión de información.

En el bus se encuentran dos pistas separadas, el bus de datos y el bus de direcciones. La CPU escribe la dirección de la posición deseada de la memoria en el bus de direcciones accediendo a la memoria, teniendo cada una de las líneas carácter binario. Es decir solo pueden representar 0 o 1 y de esta manera forman conjuntamente el número de la posición dentro de la memoria (es decir: la dirección)

Estructuras de interconexión

Existen dos estructuras de operaciones Entrada/Salida que tienen que ver con los buses, son: el bus único y el bus dedicado.

El bus dedicado trata a la memoria de manera distinta que a los periféricos (utiliza un bus especial) al contrario que el bus único que los considera a ambos como posiciones de memoria. Este bus especial que utiliza el bus dedicado tiene 4 componentes fundamentales:

- Datos: Intercambio de información entre la CPU y los periféricos.
- Control: Lleva información referente al estado de los periféricos (petición de interrupciones).
- Direcciones: Identifica el periférico referido.
- Sincronización: Temporiza las señales de reloj.

La mayor ventaja del bus único es su simplicidad de estructura que le hace ser más económico, no permite que se realice a la vez transferencia de información entre la memoria y el procesador y entre los periféricos y el procesador.

El bus dedicado es mucho más flexible y permite transferencias simultáneas. Su estructura es más compleja y por tanto sus costes son mayores.

* EL BUS XT y EL BUS ISA (AT)

En 1980 IBM fabricó su primer PC. En ese PC colocó un bus de expansión XT que funcionaba a una velocidad de 4.77 Mhz. El ancho de banda de ese bus era de 8 bits. Posteriormente se diseñó el bus AT, que en relación con el bus de datos, tenía finalmente 16 bits (ISA), Era compatible con su antecesor. También se amplió el bus de direcciones, concretamenta hasta 24 bits. Se aumentó la velocidad de cada una de las señales de frecuencia, de manera que toda la circulación de bus se desarrollaba más rápidamente. De 4.77 Mhz en el XT se pasó a 8.33 Mhz . Tenía una velocidad de transferencia máxima de 8 Mbps.

*** BUS MICRO CHANNEL (MCA)**

El diseño MCA (Micro Channel Arquitecture) permitía una ruta de datos de 32 bits, más ancha y una velocidad de reloj ligeramente más elevada de 10 Mhz, con una velocidad de transferencia máxima de 20 Mbps. Pero lo que es más importante el novedoso diseño de bus de IBM incluyó un circuito de control especial a cargo del bus, que le permitía operar independientemente de la velocidad e incluso del tipo del microprocesador del sistema.

El inconveniente era que la arquitectura de IBM era totalmente incompatible con las tarjetas de expansión que se incluyen en el bus ISA. Esto se debía a que los conectores de las tarjetas de expansión MCA eran más pequeños que las de los buses ISA.

*** EISA (Extended ISA)**

El principal rival del bus MCA fue el bus EISA (1987-1988), que también estaba basado en la idea de controlar el bus desde el microprocesador y ensanchar la ruta de datos hasta 32 bits. EISA mantuvo compatibilidad con las tarjetas de expansión ISA ya existentes lo cual le obligó a funcionar a una velocidad de 8.33 Mhz.

En una máquina EISA, puede haber al mismo tiempo hasta 6 buses principales con diferentes procesadores centrales y con sus correspondientes tarjetas auxiliares.

En este bus hay un chip que se encarga de controlar el tráfico de datos señalando prioridades para cada posible punto de colisión o bloqueo mediante las reglas de control de la especificación EISA.

Este chip actúa en la CU como un controlador del tráfico de datos. A pesar de ser mejor que los anteriores, este bus no llegó a sustituirlos porque los sistemas que disponían de él eran de mayor coste y en aquellos tiempos no se necesitaba más velocidad de la que los otros ofrecían.

*** LOCAL BUS**

Al contrario que con el EISA, MCA y PCI, el bus VL no sustituye al bus ISA sino que lo complementa. (se acopla directamente en la CPU)

Un PC con bus VL dispone para ello de un bus ISA y de las correspondientes ranuras para tarjetas de ampliación. En un Pc con Bus VL puede haber una, dos e incluso tres ranuras de expansión

El VL es una expansión homogeneizada de bus local, que funciona a 32 bits, pero que pude realizar operaciones a 16 bits. VESA presentó la primera versión del estándar VL-BUS en agosto de 1992. La aceptación por parte del mercado fue inmediata. La velocidad con este tipo de buses depende del número de conectores, cuanto más hay mayor es la velocidad. La mejor combinación de rendimiento y funciones se da a 33 Mhz.

Años más tarde se creó la versión 2.0 Este bus funcionaba a 64 bits y además mantenía toda la compatibilidad con el VL-BUS. Constaba hasta de 3 ranuras a 40 Mhz y dos a 50 Mhz.

*** PCI (Peripheral Component Interconnect)**

(Este es el bus que se usa en la actualidad) El bus PCI, (interconexión de los componentes periféricos) es independiente de la CPU, ya que entre la CPU y el bus PCI se instalará siempre un controlador de bus PCI. El bus PCI no depende del reloj de la CPU, porque está separado de ella por el controlador del bus.

El límite práctico en la cantidad de conectores para buses PCI es de tres; como ocurre, con el VL. El actual estándar PCI autoriza frecuencias de reloj que oscilan entre 20 y 33 Mhz.

ENTRADA Y SALIDA (E/S)

Funciones que realiza

Funciones que debe realizar un computador para ejecutar trabajos de entrada/salida:

- Direccionamiento o selección del dispositivo que debe llevar a cabo la operación de E/S.
- Transferencia de los datos entre el procesador y el dispositivo (en uno u otro sentido).
- Sincronización y coordinación de las operaciones.

Esta última función es necesaria debido a la diferencia de velocidades entre los dispositivos y la CPU y a la independencia que debe existir entre los periféricos y la CPU.

Una transferencia elemental de información es la transmisión de una sola unidad de información (normalmente un byte) entre el procesador y el periférico o viceversa. Para efectuar una transferencia elemental de información son precisas las siguientes funciones:

- Comunicación física entre el procesador y el periférico para la transmisión de la unidad de información.
- Control de los periféricos. Para realizar estas funciones la CPU gestionará las líneas de control necesarias.

Una operación de E/S es el conjunto de acciones necesarias para la transferencia de un conjunto de datos. Para la realización de una operación de E/S se deben efectuar las siguientes funciones:

- Recuento de las unidades de información transferidas (normalmente bytes) para reconocer el fin de operación.
- Sincronización de velocidad entre la CPU y el periférico.
- Detección de errores (e incluso corrección).
- Almacenamiento temporal de la información. Es más eficiente utilizar un buffer temporal específico para las operaciones de E/S que utilizan el área de datos del programa.
- Conversión de códigos, conversión serie/paralelo, etc.

Dispositivos externos

Una de las funciones básicas del computador es comunicarse con los dispositivos exteriores, es decir, el computador debe ser capaz de enviar y recibir datos desde estos dispositivo. Sin esta función, el ordenador no sería operativo porque sus cálculos no serían visibles desde el exterior.

De todos los posibles periféricos, algunos son de lectura, otros de escritura y otros de lectura y escritura. Por otra parte, existen periféricos de almacenamiento también llamados memorias auxiliares o masivas.

La mayoría de los periféricos están compuestos por una parte mecánica y otra parte electrónica. A la componente electrónica del periférico se le denomina controlador del dispositivo o adaptador del dispositivo. Si el dispositivo no tiene parte mecánica, el controlador estará formado por la parte digital del circuito. Frecuentemente los controladores de los dispositivos están alojados en una placa de circuito impreso diferenciada del resto del periférico.

El mayor inconveniente que encontramos en los periféricos es la diferencia entre sus velocidades de transmisión y la diferencia entre éstas y la velocidad de operación del computador.

Uso de interrupciones

Un computador debe disponer de los elementos suficientes para que el programador tenga un control total sobre todo lo que ocurre durante la ejecución de su programa. La llegada de una interrupción provoca que la CPU suspenda la ejecución de un programa e inicie la de otro.

Tipos

- Dispositivos de entrada

Estos dispositivos permiten al usuario del ordenador introducir datos, comandos y programas en la CPU.El teclado es un dispositivo de entrada. La información introducida con el mismo, es transformada por el ordenador en modelos reconocibles por el mismo. Otros dispositivos de entrada son los lápices ópticos, joysticks y el ratón, que convierte el movimiento físico en movimiento dentro de una pantalla de ordenador; los escáneres luminosos, y los módulos de

reconocimiento de voz. Otros dispositivos de entrada, usados en la industria, son los sensores.

- Dispositivos de Entrada/Salida

Los dispositivos de almacenamiento externo más frecuentes son los disquetes y los discos duros, aunque la mayoría de los grandes sistemas informáticos utiliza bancos de unidades de almacenamiento en cinta magnética. Los discos flexibles pueden contener, desde varios centenares de miles de bytes hasta bastante más de un millón de bytes de datos. Los discos duros no pueden extraerse de los receptáculos de la unidad de disco, que contienen los dispositivos electrónicos para leer y escribir datos sobre la superficie magnética de los discos y pueden almacenar desde varios millones de bytes hasta algunos centenares de millones. La tecnología de CD-ROM, que emplea las mismas técnicas láser utilizadas para crear los discos compactos de audio, permiten capacidades de almacenamiento del orden de varios cientos de megabytes de datos. También hay que añadir los recientemente aparecidos DVD que permiten almacenar más de 4 Gb de información.

- Dispositivos de salida

Estos dispositivos permiten al usuario ver los resultados. El monitor presenta los caracteres y gráficos en una pantalla similar a la del televisor. Por lo general, los monitores tienen un tubo de rayos catódicos, aunque los ordenadores pequeños y portátiles utilizan hoy pantallas de cristal líquido o electroluminiscentes. Otros dispositivos de son las impresoras, las tarjetas de sonido y los módem. Un módem enlaza dos ordenadores transformando las señales digitales en analógicas para que los datos puedan transmitirse a través de las líneas telefónicas convencionales.

PERIFÉRICOS Y CONEXIONES A DISPOSITIVOS EXTERNOS

TECLADO

El teclado es un componente al que se le da poca importancia, especialmente en los ordenadores clónicos. Si embargo es un componente esencial, pues es el que permitirá que nuestra relación con el ordenador sea fluida y agradable, de hecho, junto con el ratón son los responsables de que podamos interactuar con nuestra máquina.

Así, si habitualmente usamos el procesador de textos, hacemos programación, u alguna otra actividad en la que hagamos un uso intensivo de este componente, es importante escoger un modelo de calidad. En el caso de que seamos usuarios esporádicos de las teclas, porque nos dediquemos más a juegos o a programas gráficos, entonces cualquier modelo nos servirá, eso sí, que sea de tipo mecánico.

Actualmente sólo quedan dos estándares en cuanto a la distribución de las teclas, el expandido, que IBM lo introdujo ya en sus modelos AT, y el de *Windows95*, que no es más que una adaptación del extendido, al que se le han añadido tres teclas de más, que habitualmente no se usan, y que sólo sirven para acortar la barra espaciadora hasta límites ridículos.

En cuanto al conector, también son dos los estándares, el *DIN*, y el *mini-DIN*. El primero es el clásico de toda la vida, y aún es el habitual en equipos clónicos. El segundo, introducido por IBM en sus modelos *PS/2,* es usado por los fabricantes "de marca" desde hace tiempo, y es el habitual en las placas con formato *ATX*. De todas formas, no es un aspecto preocupante, pues hay convertidores de un tipo a otro. Nos dejamos otro tipo de conector cada vez más habitual, el *USB*, pero la verdad es que de momento apenas hay teclados que sigan este estándar.

PUERTOS

Los ordenadores personales actuales aún conservan prácticamente todos los puertos heredados desde que se diseñó el primer PC de IBM. Por razones de compatibilidad aún seguiremos viendo este tipo de puertos, pero poco a poco irán apareciendo nuevas máquinas en las que no contaremos con los típicos conectores serie, paralelo, teclado, etc... y en su lugar sólo encontraremos puertos USB, Fireware (IEE 1394) o SCSI.

Un ejemplo típico lo tenemos en las máquinas iMac de Apple, que aunque no se trate de máquinas PC-Compatibles, a nivel hardware comparten muchos recursos, y nos están ya marcando lo que será el nuevo PC-2000 en cuanto a que sólo disponen de bus USB para la conexión de dispositivos a baja-media velocidad, como son el teclados, ratón, unidad ZIP, módem, etc..

Tampoco hay que olvidar otro tipo de conectores que son ya habituales en los ordenadores portátiles como los puertos infrarrojos, que pueden llegar a alcanzar velocidades de hasta 4 Mbps y que normalmente cumplen con el estándar IrDA, o las tarjetas PC-Card (antiguamente conocidas como PCMCIA) ideales para aumentar la capacidad de dichas máquinas de una manera totalmente estándar.

USB

Desde que nació el PC de la mano de I.B.M., por motivos de compatibilidad, algunas de sus características han permanecido inalterables al paso del tiempo.

Conectores como el de la salida paralelo (o Centronics), la salida serie (RS-232) o el conector del teclado han sufrido muy pocas variaciones.

Si bien es cierto que estos conectores todavía hoy cumplen su función correctamente en casos como la conexión de un teclado, un ratón o un modem, se han quedado ya desfasados cuando tratamos de conectar dispositivos más rápidos como por ejemplo una cámara de video digital.

USB nace como un estandar de entrada/salida de velocidad media-alta que va a permitir conectar dispositivos que hasta ahora requerían de una tarjeta especial para sacarles todo el rendimiento, lo que ocasionaba un encarecimiento del producto además de ser productos propietarios ya que obligaban a adquirir una tarjeta para cada dispositivo.

Pero además, USB nos proporciona un único conector para solventar casi todos los problemas de comunicación con el exterior, pudiendose formar una auténtica red de periféricos de hasta 127 elementos.

Mediante un par de conectores USB que ya hoy en día son estandar en todas las placas base, y en el espacio que hoy ocupa un sólo conector serie de 9 pines nos va a permitir conectar todos los dispositivos que tengamos, desde el teclado al modem, pasando por ratones, impresoras, altavoces, monitores, scaners, camaras digitales, de video, plotters, etc... sin necesidad de que nuestro PC disponga de un conector dedicado para cada uno de estos elementos, permitiendo ahorrar espacio y dinero.

Al igual que las tarjeta ISA tienden a desaparecer, todos los conectores anteriormente citados también desaparecerán de nuestro ordenador, eliminando además la necesidad de contar en la placa base o en una tarjeta de expansión los correspondientes controladores para dispositivos serie, paralelo, ratón PS/2, joystick, etc...

Como podeis ver, realmente es un estándar que es necesario para facilitarnos la vida, ya que además cuenta con la famosa característica PnP (Plug and Play) y la facilidad de conexión "en caliente", es decir, que se pueden conectar y desconectar los periféricos sin necesidad de reiniciar el ordenador.

Otras características que también deberemos saber son:

* Dos velocidades de acceso, una baja de 1,5 Mbps para dispositivos lentos como pueden ser joysticks o teclados y otra alta de 12 Mbps para los dispositivos que necesiten mayor ancho de banda.

* Topología en estrella, lo que implica la necesidad de dispositivos tipo "hub" que centralicen las conexiones, aunque en algunos dispositivos como teclados y monitores ya se implementa esta característica, lo que permite tener un sólo conector al PC, y desde estos dispositivos sacar conexiones adicionales. Por ejemplo en los teclados USB se suele implementar una conexión adicional para el ratón, o incluso otras para joystick, etc.. y en los monitores varias salidas para el modem, los altavoces...

* Permite suministrar energía electrica a dispositivos que no tengan un alto consumo y que no estén a más de 5 metros, lo que elimina la necesidad de conectar dichos periféricos a la red eléctrica, con sus correspondientes fuentes de alimentación, como ahora ocurre por ejemplo con los modems externos.

* En los ordenadores Mac más modernos (como el iMAC) también están implementados dichos conectores, lo que da una idea de su estandarización, y redundará en favor de una mayor gama de productos y mejor competitividad.

* Si trabajamos bajo Windows necesitaremos como mínimo la versión OSR 2.1 del Windows 95 para que reconozca los dispositivos.

PUERTO PARALELO

Tras la acentuada falta de estandarización del interfaz paralelo, surgió Centronics como un standard en este tipo de conexión, debido a la facilidad de uso y la comodidad a la hora de trabajar con él.

A raíz de este interfaz, posteriormente apareció una norma standard (IEEE 1284) para el interfaz paralelo en los ordenadores personales, en la cual se tratan varios tipos de protocolos los cuales se verán a lo largo de este trabajo.

La transmisión en paralelo entre un ordenador y un periférico, se basa en la transmisión de datos simultáneamente por varios canales, generalmente 8 bits. Por esto se necesitan 8 cables para la transmisión de cada bit, mas otros tantos cables para controles del dispositivo, el numero de estos dependerá del protocolo de transmisión utilizado.

Los principales tipos y nombres de canales que son utilizados como control son:

- ***STROBE*** - a través de el, el ordenador comunica al periférico que esta preparado para transmitir.

- ***BUSY*** - el periférico comunica a través de el, que NO esta preparado para recibir datos.

- ***ACK*** - el periférico comunica a través de el, que esta preparado para recibir datos.

- ***SELECT Y SELECTIN*** - indican el tipo de error producido en el periférico.
- ***ERROR*** - indica que se ha producido un error en el periférico.
- ***PE*** - depende del tipo del periférico, en el caso de la impresora indica que no tiene papel.

Algunos de estos canales pueden ser utilizados para alguna acción adicional o cambiar la anteriormente descrita, según el protocolo que se utilice.

PUERTO SERIE

El puerto serie de un ordenador es un adaptador asíncrono utilizado para poder intercomunicar varios ordenadores entre sí.

Un puerto serie recibe y envía información fuera del ordenador mediante un determinado software de comunicación o un driver del puerto serie.

El software envía la información al puerto carácter a carácter, conviertiéndolo en una señal que puede ser enviada por un cable serie o un módem.

Cuando se ha recibido un carácter, el puerto serie envía una señal por medio de una interrupción indicando que el carácter está listo. Cuando el ordenador ve la señal, los servicios del puerto serie leen el carácter.

TIPOS DE INSTALACIÓN

En este punto puede elegir entre la actualización de un sistema ESware, ya sea una versión anterior o una reparación de la versión actual, o instalar un sistema nuevo.

La actualización realizará una comprobación de los paquetes instalados y los sustituirá con nuevas versiones; esto incluye todo el sistema X Window, *kernel* y demás partes base del sistema. Los archivos de configuración serán salvaguardados en la medida de lo posible, creando copias de seguridad de los mismos con extensión .rpm .

Tenemos cuatro alternativas en la instalación de un nuevo sistema : una instalación personalizada, que nos da el máximo control (recomendada), y tres instalaciones más automatizadas (estación de trabajo, servidor y portátiles).

- Instalación personalizada: configuraremos las particiones manualmente y decidiremos qué paquetes serán instalados y cuáles no.

- Instalación para estación de trabajo: este tipo de instalación está indicado en sistemas con particiones Linux ya creadas (de un sistema anterior). Aprovechará las particiones ya creadas para instalar un nuevo sistema, con la consecuente pérdida de datos de las mismas. Una selección concreta de paquetes será instalada automáticamente, ocupando un espacio aproximado de 500 MB.

- Instalación para servidor: es la instalación más automatizada. Borrará toda la información de nuestro disco (o discos) y realizará un particionado del mismo a su conveniencia, realizando una instalación de paquetes, también automatizada, con herramientas habituales en todo servidor de red.

- Instalación para portátiles: es similar a la instalación para una estación de trabajo, cambiando la selección de paquetes, en favor de un sistema portátil.

Es bueno recalcar y marcar, de nuevo, la diferencia entre dos posibles escenarios (puede haber muchos más, pero ...):

- Un entorno de test o no excesivamente sensible, según el cual no es necesarios crear un entorno especialmente sólido.

- Un entorno de trabajo: servidor, estación de trabajo, el cual debemos procurar sea lo más sólido y seguro posible.

- Por esta razón, en el segundo de los escenarios, debemos intentar una instalación compacta que incluya solo aquellos componentes necesarios ....

NECESIDADES DE PARTICIONADO DE LINUX

Linux requiere por lo menos una partición, para el sistema de archivos raíz. Si desea crear varios sistemas de archivos, necesitará una partición por cada sistema de archivos. Por lo general, se crearán, al menos, dos particiones para Linux: una para ser usada como sistema de archivos raíz, y la otra como espacio de intercambio. Por supuesto, hay otras opciones .

> El tamaño de las particiones de su sistema Linux depende en gran parte de qué software quiera instalar en él. El tamaño de sus particiones de intercambio (debe elegir una para esto) depende de la "RAM virtual" que necesite.

Las razones para crear diferentes particiones particiones:

- una partición aloja un Sistema de ficheros (*filesystem*)

- sirven para aislar partes sensibles de nuestro sistema y, de esta forma, asegurar que el fallo de una parte no afecta a otra

- permiten aplicar opciones a unas particiones y no a otras, p.ej.: una partición HOME puede tener activadas las quotas de disco, otra puede ser de solo lectura, etc ...

PROBLEMAS CON DISCOS DUROS GRANDES

Debido a las limitaciones de algunas BIOS, habitualmente no es posible arrancar desde particiones que empiecen más allá del cilindro 1024. Esto ha cambiado con las últimas versiones de LILO, el gestor de arranque de Linux.

Pero, aunque ha cambiado debemos prestar atención a ello y crear la partición ***/boot*** para aliviar ese problema y no tener que preocuparnos. Si existe la partición ***/boot*** (existirá si la hemos creado al principio del disco), se escribirán en ella por defecto los contenidos del directorio /boot (es decir, todos los archivos que LILO necesita para arrancar el sistema). Al arranque el sistema montará esta partición sobre el directorio /boot y LILO podrá acceder a estos archivos que de esta forma estarán por encima del cilindro 1024 aunque este hecho sea transparente al usuario.

DISPOSITIVOS Y PARTICIONES EN LINUX

Muchas distribuciones necesitan que se creen a mano las particiones de Linux utilizando el programa fdisk. Otras pueden crearlas durante el mismo proceso de instalación. En cualquier caso, debemos conocer lo siguiente acerca de los nombres para los dispositivos y las particiones en Linux. Bajo Linux, los dispositivos y las particiones tienen nombres muy distintos a los utilizados en otros sistemas operativos. En MS-DOS / Windows, las disquetes se identifican como A: y B:, mientras que las particiones del disco duro se identifican como C:, D:, etc. Con Linux, la denominación es algo diferente .

Dispositivo (en MS-DOS, ...)	***En Linux***
Primera disquetera (A: en MS-DOS)	/dev/fd0
Segunda disquetera (B: en MS-DOS)	/dev/fd1
Primer disco duro (todo el disco)	/dev/hda
Primer disco duro, partición primaria 1	/dev/hda1
Primer disco duro, partición primaria 2	/dev/hda2
Primer disco duro IDE, partición lógica 1	/dev/hda5
Primer disco duro IDE, partición lógica 2	/dev/hda6
Segundo disco duro IDE (todo el disco)	/dev/dB
Segundo disco duro IDE, partición primaria 1	/dev/hdb1
Primer disco duro SCSI (todo el disco)	/dev/sda
Primer disco duro SCSI, partición primaria 1	/dev/sda1
Segundo disco duro SCSI (todo el disco)	/dev/sdb
Segundo disco duro SCSI, partición primaria 1	/dev/sdb1

Observe que /dev/fd0 corresponde a la primera disquetera (A: bajo Windows) y que /dev/fd1 es la segunda (B:). Los discos duros SCSI se nombran de manera diferente a otros discos. Los IDE, EIDE y CD-ROM (que no sean SCSI) se acceden a través de /dev/hda, /dev/hdb, etc. Las particiones de /dev/hda son /dev/hda1, /dev/hda2, etc. Los dispositivos SCSI son /dev/sda, /dev/sdb, etc., y las particiones con /dev/sda1, /dev/sda2, etc. Las particiones lógicas se nombran de forma consecutiva partiendo de /dev/hda5.

PARTICIONADO AUTOMÁTICO

El particionado automático ocasionará la pérdida de los datos existentes en el disco duro; una pantalla de aviso nos indicará de esta circunstancia. No obstante, las instalaciones para estación de trabajo, servidor y portátil también permiten la configuración manual del disco.

PARTICIONADO DEL SISTEMA

La elección de las particiones de nuestro sistema es un paso importante en el proceso de instalación. El número mínimo de particiones a realizar es dos: una partición para el sistema Linux (partición Linux Native) y una partición de intercambio (Linux Swap). Es necesario indicar en qué partición se instalará el sistema estableciendo el punto de montaje como /. Para el resto de las particiones (salvo la *swa*p) indicaremos los puntos de montaje que deseemos. Por ejemplo, si tenemos una partición msdos y queremos acceder a ella desde ESware Linux la podemos montar en /dos.

GESTORES DE PARTICIONES

Como ya hemos aprendido, para poder instalar el sistema operativo nos hace falta tener particionado nuestro disco duro, para poder situar dentro de estas los sistemas de archivos. Hay varios gestores de particiones dentro de GNU Linux, pero tenemos que destacar tres.

El primero es fdisk. Esta aplicación esta portada a otros sistemas operativos, como por ejemplo windows. Con élla podremos crear particiones, borrarlas y darles formato. El segundo es cfdisk un front end gráfico para fdisk. La tercera aplicación es **parted**, para usuarios avanzados,también con un interface gráfico, **nparted** que le dá un aspecto mucho más agradable.

CREACIÓN DE LOS SISTEMAS DE ARCHIVO (FILE-SYSTEM)

Antes de que se puedan usar las particiones de Linux para almacenar ficheros, hay que crear los sistemas de archivos en ellas. La creación de un sistema de archivos es análoga a formatear una partición en Windows u otros sistemas operativos.

El tipo de sistema de archivos más usado es el Sistema de Ficheros Extendido 2, o ext2fs. El ext2fs es uno de los sistemas más eficientes y flexibles; permite hasta 256 caracteres en los nombres de los ficheros y tamaños de estos de hasta 4 Terabytes .

De nuevo, hablar del formato ***reiserfs*** para sistemas de ficheros (*filesystems*), que es muy fiable, seguro, etc. ....

El propio proceso de instalación crea los sistemas de archivos de forma automática. Si desea crear sus propios sistemas a mano, siga el método que a

continuación describimos.

Lo que viene a continuación muestra la forma de crear un *filesystem* desde linea de comando, por lo tanto, hay que comentar que el párrafo anterior es como se realiza durante la instalación y que las líneas siguientes se refieren a los pasos a seguir para crear *filesystems* en un sistema en funcionamiento. Donde digo ***mke2fs***, también podría decir: ***mkfs*** o ***mkreiserfs***

Para crear un sistema de tipo ext2fs utilice el comando:

mke2fs [-c] <partición> [tamaño]

donde <partición> es el nombre de la partición, y <tamaño> es el tamaño de la partición en bloques. Aquí, la opción <tamaño> actúa igual que con mkswap;, si se omite el tamaño, se obtiene automáticamente. Por ejemplo, para crear un sistema de 82080 bloques en /dev/hda2, use el comando:

 # mke2fs [-c] /dev/hda2 82080

o la versión más sencilla:

 # mke2fs [-c] /dev/hda2

Si quiere usar varios sistemas de archivos en Linux, necesitará repetir el comando mke2fs por cada sistema de archivos. Esto funciona exactamente igual para los disquetes y nuevos discos duros que se añadan posteriormente al sistema.

CREACIÓN DEL ESPACIÓ DE INTERCAMBIO (SWAP)

La información que viene a continuación no es necesaria durante el proceso de instalación, ya que se hace todo automáticamente. Si después de tener instalado el sistema queremos (o necesitamos) cambiar la partición de intercambio, aquí tenemos los pasos que debemos dar.

El comando utilizado para preparar una partición de intercambio es **mkswap**, cuya sintaxis es:

mkswap [-c] <partición> [tamaño]

donde <*partición*> es el nombre de la partición de intercambio y <*tamaño*> es el tamaño de la partición, en bloques. Si el tamaño se omite, mkswap lo asigna automáticamente; es decir, que si al hacer la partición ya definimos el tamaño correctamente, será más cómodo omitir el tamaño. Por ejemplo, si su partición de intercambio es la /dev/hda3 y tiene 10336 bloques, teclee el comando:

 # mkswap -c /dev/hda3 10336

o la versión más sencilla:

 # mkswap -c /dev/hda3

La opción *-c* indica a mkswap que compruebe si hay bloques erróneos en la partición mientras la crea.

Si se usan varias particiones de intercambio, se necesitará ejecutar el comando mkswap apropiado para cada partición.

Después de preparar el área de intercambio, hay que decirle al sistema que la use. Normalmente, el sistema comienza a usarla automáticamente durante el arranque.

El comando para hacerlo es swapon, y tiene el siguiente formato:

swapon <partition>

En el ejemplo anterior, para activar el espacio de intercambio en /dev/hda3, usaremos el comando:

 # swapon /dev/hda3

FORMATEADO DE LOS DISCOS

El formateo de particiones implica la pérdida de los datos de las mismas. Esto es necesario en el caso d**e** particiones recién creadas y puede ser interesante para instalaciones nuevas. (**mkfs, mke2fs, mkreiserfs .. mformat a:)**

INSTALACIÓN DEL LILO

Lilo *(Linux Loade*r) es un excelente gestor de arranque que nos permite seleccionar el sistema operativo con el que iniciar cada vez la máquina. El lugar habitual para instalarlo es el MBR *(Master Boot Recor*d); si utiliza otro gestor, lo puede instalar en el primer sector de la partición de arranque. Debe indicar los sistemas que desea incluir en el arranque poniendo las correspondientes etiquetas a cada uno de ellos.

CONFIGURACIÓN DE LA RED

Si tiene una red de área local, debe configurar los parámetros:

1. Dirección IP.

Es la dirección de red del ordenador.

El rango de direcciones reservadas para LAN es de 192.168.0.0 a 192.168.255.255, usar una dirección de este rango le garantiza que no tendrá conflictos si se conecta a Internet desde esta red. Si no conoce las direcciones que se usan en su red deberá ponerse en contacto con el administrador del sistema.

2. Mascara de red.

La mascara de red nos indica la cantidad máxima de equipos que puede estar conectados a nuestra red. Hay tres clases:

- Clase A: x.0.0.0
- Clase B: x.x.0.0
- Clase C: x.x.x.0

3. Dirección de Red.

Es la dirección de la red en la que esta conectado el ordenador.

En una red local sería 192.168.10.0

4. Dirección de Difusión (Broadcast)

Es la dirección de red a la cual se mandan mensajes para que todo el que este en nuestra red se entere.

5. Nombre del Host

El nombre del host es el nombre de nuestro ordenador en la red, formado por el *nombre .dominio.com*.

6. Puerta de enlace (Gateway)

Es la dirección a través de la cual salimos de nuestra red local hacia el exterior.

Para poder usar DHCP (asignación dinámica de direcciones IP) necesita una máquina en su red que esté configurada como servidor DHCP.

7. DNS Primario, DNS Secundario

aquí pondremos las direcciones de los servidores DNS que no sirve para localizar los ordenadores en nuestra red.

CONFIGURACIÓN DE LA ZONA HORARIA

Por defecto, la configuración horaria está preparada para Europa/Madrid. Si vd no esta instalando el sistema en la España, lógicamente debe cambiar la zona horaria. Esto es ta basado en los usos horarios internacionales así pues a Madrid le corresponde UTC +1 (GMT+1) y si se encuentra en las Islas Canarias le corresponde UTC.

CONTRASEÑA DE ROOT Y CUENTAS DE USUARIO

La contraseña de *root* es la más importante del sistema. Es obligado introducir más de seis caracteres y procure no usar palabras que estén en diccionarios para dificultar cualquier posible ataque. Un ataque consiste en un intento de capturar las claves de los usuarios del sistema, existiendo programas que se usan

para *crackear* estas claves. Desde esta pantalla puede, además, agregar cuentas de otros usuarios.

Nota importante. Recuerde que no debe usar la cuenta de root habitualmente. Añada una cuenta de usuario y sólo entre como root cuando realice tareas administrativas.

CONFIGURACIÓN DE LA AUTENTICACIÓN

El sistema MD5 emplea una encriptación más fuerte para almacenar las contraseñas (recomendado). Si decide utilizar ***Shadow password*** (también recomendado) tendrá una medida de seguridad añadida en su sistema.

S*hadow password* es una forma de intentar dar más seguridad a las claves, mediante un método de encriptación MD5.

Únicamente deberá activar NIS *(Network Infomation Servic*e) si está trabajando en una red en la que los puestos de trabajo se autentifican ante un servidor NIS.

CONFIGURACIÓN DE LA IMPRESORA

Aquí se define que impresora vamos a utilizar.

Las impresoras locales son las que tenemos conectadas a nuestro ordenador en el puerto paralelo, lp0, lp1, lp2 (lpt1, lpt2, de otros OS). Debe de saber el modelo exacto para poderlo elegir en la lista y el lugar donde será almacenada la cola de impresión.

Las impresoras remotas son aquellas a las cuales accedemos a través de la red local, debemos saber cual es su dirección ip, el equipo al que están conectadas. Si el servidor de impresión es Windows o Linux con SAMBA debe activar la opción usar SAMBA y especificar el nombre de el recurso compartido.

SELECCIÓN DE GRUPOS DE PAQUETES

Tiene opción de seleccionar los paquetes que se van a instalar, que son lo siguientes

1. Base. Es la base del sistema

2. Soporte Impresión. Software para poder imprimir

3. Sistema X Window. Es la base del entorno gráfico.

4. Configuración PNP Lothar. Nos permite configurar dispositivo Plug and Play (PNP), con un solo click del ratón.

5. KDE. Gestor de ventanas, para el entorno gráfico

6. KDE Desarrollo. Entorno de desarrollo para KDE.

7. GNOME. Otro entorno gráfico.

8. Herramientas Correo/WWW/Noticias. Conjunto de programas para trabajar en internet . .

9. Conectividad DOS/Windows. Paquetes para poder trabajar con sistemas DOS/Windows.

10. Manipulación Gráfica. Programas para manipular ficheros gráficos.

11. Juegos. Juegos para entorno gráfico

12. Soporte Multimedia. Software, sonido y vídeo.

13. Especial Portátiles Software especifico para ordenadores portátiles

14. Estación de Trabajo con Soporte de Red. Software para la conexión a redes locales.

15. Estación de Trabajo con Soporte Modem. Software para la conexión a redes locales vía modem.

16. Servidor de Red. Programas para que funcione el ordenador como servidor en una LAN.

17. Servidor de Correo Sendmail. Programa para que funcione como servidor de correo.

18. Servidor de Correo Postfix. Programa para que funcione como servidor de correo.

19. Servidor News. Programa para que funcione como servidor de grupos

de noticias.

20. Servidor NFS. Programa para que funcione como servidor de ficheros NFS.

21. Servidor Samba. Programa para que funcione como servidor para clientes Windows

22. Servidor SSH. Programa para que funcione como servidor de conexiones seguras.

23. Servidor POP3. Programa para que funcione como servidor de correo POP3.

24. Servidor FTP. Programa para que funcione como servidor de ficheros.

25. Servidor Web. Programa para que funcione como servidor de páginas web.

26. Servidor Web Seguro. Programa para que funcione como servidor de páginas web seguro.

27. Servidor de Nombres. Programa para que funcione como servidor de DNS.

28. Servidor Bases de Datos Postgres. Servidor de Bases de Datos.

29. Servidor Bases de Datos MySQL. Servidor de Bases de Datos**.**

30. Estación Administradora de Red. Software de control y administración de red.

31. Administrador de Servidores Webmin.

32. Extras de Seguridad. Programas extras para seguridad.

33. Autoedición y Publicación. Software de Diseño Gráfico.

34. Emacs . Editor Programable.

35. Desarrollo. Software para programar.

36. Desarrollo del Kernel . Software para programar el kernel..

37. Clustering. Herramientas de alta disponibilidad.

38. Utilidades.

39. Workstation Commom. Programas comunes para redes.

40. GNOME Workstation. Cliente de red para gnome

41. KDE Workstation. Cliente de red para kde

SELECCIÓN DE PAQUETES INDIVIDUALES

Una vez seleccionados los paquetes, podrá decidir qué software específico se instalará (Sólo si se activó la opción ***Seleccionar paquetes individualmente***.) Esto, no es ni más ni menos, es instalar programas individualmente, indicando que programas se instalan y cuales no, independientemente de su relación con los paquetes anteriores.

DEPENDENCIAS SIN RESOLVER

Se presentará una pantalla de dependencias no resueltas si ha seleccionado algún paquete que requiere tener otro instalado para su correcto funcionamiento. Pulse ***Aceptar***. Debe tener en cuenta que la instalación no es solo pulsar aceptar (así es como funciona Windows y Mac) y confiar nos aleja de saber lo que está pasando: CUIDADO CON ESTO.

CONFIGURACIÓN PERSONALIZADA DEL ENTORNO GRÁFICO

Aquí puede probar la configuración del servidor de ventanas y decidir si quiere entrar al sistema en modo gráfico directamente . Si la configuración automática no es de su agrado, puede retocarla hasta dejarla a su gusto. Para ello hay que seleccionar la opción ***Personalizar Configuración***. Lea la parte del manual dedicada a la configuración de X Window si no tiene mucha experiencia.

CREACIÓN DEL DISCO DE ARRANQUE

¡Altamente recomendable ! Quizá no llegue a usarlo, pero si llegase a tener algún problema en su sistema, es posible que le permita recuperarlo

INSTALACIÓN TERMINADA

Puede reiniciar el sistema y empezar a trabajar/disfrutar de su nuevo y flamante ESware GNU/Linux.

VARIANTES DE LA INSTALACIÓN

Ademas de la instalación estándar que acabamos de explicar, el sistema operativo Linux también se puede instalar a través de un disquete y a través de una red.

El caso de la red es el más completo. Dado este caso tendremos las siguientes variantes:

1. **NFS Image:** Tenemos en la red un ordenador compartido mediante NFS.

2. **FTP:** A través de un FTP anónimo podremos instalar el sistema operativo.

3. **HTTP:** Parecido al caso anterior, solo que en esta ocasión nos conectamos a un servidor HTTP.

4. **Disco Duro**: La última posibilidad es a través de un disco duro que este particionado en Windows, en el cual copiamos el CD completo.

DETECCIÓN Y CONFIGURACIÓN DEL HARDWARE SOPORTADO

En estos momentos tenemos que tener en cuenta una cosa que no podremos olvidar en ningún momento. El hardware en Linux esta soportado por el Kernel. Así, para que puedan ser detectados casi todos los componentes hardware, estos tienen que estar compilados dentro de las opciones del kernel.

En el capitulo de compilación del kernel aprenderemos como se tiene que compilar para que se puedan ir introduciendo todas estas novedades de hardware, que van apareciendo en cada versión nueva.

ADMINISTRACIÓN DEL SISTEMA Y USUARIOS

Un sistema Linux es grande y complicado. Y la principal tarea del administrador es que todo funcione. El administrador gestionará los recursos del sistema para que estén disponibles a los usuarios. Será necesario identificar dichos recursos y asignarlos correctamente definiendo correctamente las formas de acceso, los propietarios de los mismos y la autoridad relacionada con cada uno de ellos.

FICHEROS Y DIRECTORIOS

Un fichero es un conjunto de información al que se le ha asignado un nombre (llamado nombre del fichero).

Ejemplos de fichero son un mensaje de correo, o un programa que puede ser ejecutado. Esencialmente, cualquier cosa salvada en el disco es guardada como un fichero individual.

Los ficheros son identificados por sus nombres. Estos nombres usualmente identifican el fichero y su contenido de alguna forma significativa para usted. Las extensiones del tipo: fichero.tar, fichero.jpg no tienen vinculación alguna con el sistema de ficheros (FS) y por lo tanto, no son tenidas en cuenta por el núcleo de Linux, que identifica a cada fichero por un número de inodo único. Pero estas extensiones "asocian" al fichero.tar.gz, con aplicaciones como " tar " o " gzip " o a un fichero.jpg con un programa de visionado o retoque de imágenes fotográficas.

No hay un formato estándar para los nombres de los ficheros como lo hay en MS-DOS y en otros sistemas operativos; en general, estos nombres pueden contener cualquier carácter (excepto /), y están limitados a 256 caracteres de longitud.

Un directorio es simplemente una colección de ficheros. Puede ser considerado como una «carpeta» que contiene muchos ficheros diferentes.

Los directorios también tienen nombre con el que los podemos identificar. Además, los directorios mantienen una estructura de árbol; es decir, los directorios pueden contener otros directorios.

Un fichero puede ser referenciado por su nombre y una ruta de acceso, conjunto constituido por su nombre, antecedido por el nombre del directorio que lo contiene.

Ruta: **directorio/fichero**

Como puede ver, el directorio y el nombre del fichero van separados por un carácter /. Por esta razón, los nombres de fichero no pueden contener este carácter. Los directorios pueden anidarse uno dentro de otro.

Ruta: **directorio_abuelo/directorio_padre/directorio_hijo/fichero**

Por tanto, la **ruta de acceso** realmente es el camino que se debe tomar para localizar a un fichero. El directorio sobre un subdirectorio dado es conocido como el directorio padre.

EL ÁRBOL DE DIRECTORIOS

Los sistemas Linux tienen una distribución de ficheros normalizada, de forma que recursos y ficheros puedan ser fácilmente localizados. Esta distribución forma el árbol de directorios, el cual comienza en el directorio /, también conocido como directorio raíz.

Directamente por debajo de / hay algunos subdirectorios importantes: /bin, /etc, /dev y /usr, entre otros. Éstos a su vez contienen otros directorios con ficheros de configuración del sistema, programas, etc.

En particular, cada usuario tiene un directorio home. Éste es el directorio en el que el usuario guardará sus ficheros. Usualmente, los directorios home de los usuarios cuelgan de /home y son nombrados con el nombre del usuario al que pertenecen. Por tanto, el directorio home de nemo es /home/nemo.

EXPLORANDO EL SISTEMA DE FICHEROS

El sistema de ficheros es la colección de ficheros y la jerarquía de directorios de su sistema.

/bin

/bin es la abreviación de binaries, o ejecutables. Es donde residen la mayoría de los programas esenciales del sistema. La mayoría (si no todos) los ficheros de /bin tienen un asterisco (*) añadido al final de sus nombres. Esto indica que son ficheros ejecutables.

/dev

Los ficheros en /dev son conocidos como controladores de dispositivo (device drivers) y se utilizan para acceder a los dispositivos del sistema y recursos, como discos duros, modems, memoria, etc.

De la misma forma que puede leer datos de un fichero, puede leerlos desde la entrada del ratón leyendo /dev/mouse o del un puerto serie leyendo /dev/ttyS0 (ttyS0 es la denominación que Linux da al primer puerto serie en otros sistemas COM1).

Los ficheros que comienzan su nombre con fd son controladores de disqueteras. fd0 es la primera disquetera, fd1 la segunda. Hay más controladores de dispositivo para disqueteras de los que hemos mencionado. Éstos representan tipos específicos de discos. Por ejemplo, fd1H1440 accederá a discos de 3.5» de alta densidad en la disquetera 1.

En este directorio tenemos una lista de algunos de los controladores de dispositivo más usados. Nótese que incluso aunque puede que no tenga alguno de los dispositivos listados, tendrá entradas en dev de cualquier forma. /dev/console hace referencia a la consola del sistema, es decir, al monitor conectado directamente a su sistema.

Los dispositivos /dev/ttyS y /dev/cua son usados para acceder a los puertos serie. Por ejemplo, /dev/ttyS0 hace referencia a COM1 bajo MS-DOS. Los dispositivos /dev/cua son callout, los cuales son usados en conjunción con un modem.

Los nombres de dispositivo que comienzan por hd acceden a discos duros. /dev/hda hace referencia a la totalidad del primer disco duro, mientras que /dev/hda1 hace referencia a la primera partición en /dev/hda.

Los nombres de dispositivo que comienzan con sd son dispositivos SCSI. Si tiene un disco duro SCSI, en lugar de acceder a él mediante /dev/hda, deberá acceder a /dev/sda. Las cintas SCSI son accedidas vía dispositivos st y los CD-ROM SCSI vía sr.

Los nombres que comienzan por lp acceden a los puertos paralelo. /dev/lp0 hace referencia a LPT1 en el mundo MS-DOS.

/dev/null es usado como «agujero negro». Cualquier dato enviado a este dispositivo desaparece. Si desea suprimir la salida por pantalla de los errores que peda causar una orden, podría enviar la salida stderr a /dev/null.

Los nombres que comienzan por /dev/tty hacen referencia a «consolas virtuales» de su sistema (accesibles mediante las teclas <Alt+F1>,<Alt+F2>, etc). /dev/tty1 hace referencia a la primera VC, /dev/tty2 a la segunda, etc.

Los nombres de dispositivo que comienzan con /dev/pty son «pseudo-terminales». Éstos son usados para proporcionar un «terminal» a sesiones remotas. Por ejemplo, si su máquina está en una red, telnet de entrada usara uno de los dispositivos /dev/pty.

/etc

/etc contiene una serie de ficheros de configuración del sistema.

Éstos incluyen /etc/passwd (la base de datos de usuarios), /etc/rc (guiones de inicialización del sistema), etc.

/sbin

/sbin se usa para almacenar programas esenciales del sistema, que usará el administrador del mismo.

/home

/home contiene los directorios home de los usuarios. Por ejemplo, /home/madre es el directorio del usuario madre. En un sistema recién instalado, no habrá ningún usuario en este directorio (a no ser que el programa instalador nos haya permitido crear usuarios durante la instalación).

/lib

/lib contiene las imágenes de las librerías compartidas. Estos ficheros contienen código que compartirán muchos programas. En lugar de que cada programa contenga una copia propia de las rutinas compartidas, éstas son guardadas en un lugar común, en /lib. Esto hace que los programas ejecutables sean menores y reduce el espacio usado en disco.

/proc

/proc es un «sistema de ficheros virtual». Los ficheros que contiene realmente residen en memoria, no en disco. Hacen referencia a varios procesos que corren en el sistema, y le permiten obtener información acerca de qué programas y procesos están ejecutándose en un momento dado.

/tmp

Muchos programas tienen la necesidad de generar cierta información temporal y guardarla en un fichero temporal. El lugar habitual para esos ficheros es /tmp

/usr

/usr es un directorio muy importante. Contiene una serie de subdirectorios que contienen a su vez algunos de los más importantes y útiles programas y ficheros de configuración usados en el sistema.

Los directorios descritos arriba son esenciales para que el sistema esté operativo, pero la mayoría de las cosas que se encuentran en /usr son opcionales para el sistema. De cualquier forma, son estas cosas opcionales las que hacen que el sistema sea útil e interesante.

/usr/X11R6

/usr/X11R6 contiene el sistema X Window si lo instala. El sistema X Window es un entorno gráfico grande y potente, el cual proporciona un gran número de utilidades y programas gráficos, mostrados en ventanas en su pantalla.

El directorio /usr/X11R6 contiene todos los ejecutables de X Window, ficheros de configuración y de soporte.

/usr/bin

/usr/bin es el almacén real de programas del sistema Linux. Contiene la mayoría de los programas que no se encuentran en otras partes como /bin. Los ejecutables de los programas instalados a partir de paquetes RPM estarán en este directorio. Los instalados a partir de archivos.tar.gz irán a /usr/local/bin

/usr/etc

Como /etc contiene diferentes ficheros de configuración y programas del sistema, /usr/etc. En general, los ficheros que se encuentran en /usr/etc/ no son esenciales para el sistema, a diferencia de los que se encuentran en /etc, que sí lo son.

/usr/include

/usr/include contiene los ficheros de cabecera para el compilador de C. Estos ficheros (la mayoría de los cuales terminan en .h, de header) declaran estructuras de datos, subrutinas y constantes usadas en la escritura de programas en C.

Los ficheros que se encuentran en /usr/include/sys son generalmente usados en la programación de Linux a nivel de sistema. Si está familiarizado con el lenguaje de programación C, aquí encontrará los ficheros de cabecera como stdio.h, el cual declara funciones como printf().

/usr/g++-include

Contiene ficheros de cabecera para el compilador de C++ (muy parecido a /usr/include).

/usr/lib

/usr/lib contiene las librerías stub y static equivalentes a las encontradas en /lib. Al compilar un programa, éste es «enlazado» con las librerías que se encuentran en /usr/lib, las cuales dirigen al programa a buscar en /lib cuando necesita el código de la librería. Además, varios programas guardan ficheros de configuración en /usr/lib.

/usr/local

/usr/local es muy parecido a /usr: contiene programas y ficheros no esenciales para el sistema, pero que hacen el sistema más divertido y excitante. En general, los programas que se encuentran en /usr/local son específicos de su sistema, esto es, el directorio /usr/local difiere bastante entre sistemas Linux. Aquí encontrará programas grandes como TEX (sistema de formateo de documentos) y Emacs (gran y potente editor), si los instala.

/usr/man

Este directorio contiene las páginas de manual. Hay un directorio para cada sección de las paginas (use la orden man man para más detalles).

/usr/src

/usr/src contiene el código fuente (programas por compilar) de varios programas de su sistema. El más importante es /usr/src/Linux, el cual contiene el código fuente del núcleo de Linux (siempre que haya instalado las fuentes).

var

/var contiene directorios que a menudo cambian su tamaño o tienden a crecer. Muchos de estos directorios solían residir en /usr, pero desde que estamos trabajando de dejarlo relativamente inalterable, los directorios que cambian a menudo han sido llevados a /var. Algunos de estos directorios son:

/var/log

/var/log contiene varios ficheros de interés para el administrador del sistema, específicamente históricos del sistema, los cuales recogen errores o problemas con el sistema. Otros ficheros guardan las sesiones de presentación en el sistema, así como los intentos fallidos.

/var/spool

/var/spool contiene ficheros que van a ser pasados a otro programa. Por ejemplo, si su máquina está conectada a una red, el correo de llegada será almacenado en /var/spool/Correo hasta que lo lea o lo borre. Artículos nuevos de las noticias tanto salientes como entrantes, pueden encontrarse en /var/spool/news, etc.

DIRECTORIO DE TRABAJO ACTUAL

En cualquier momento, las órdenes que teclee al intérprete de comandos son dadas en términos de su directorio de trabajo actual. Puede pensar en su directorio actual de trabajo como en el directorio en el que actualmente está «situado».

Cuando entra en el sistema, su directorio de trabajo es su directorio home: /home/madre o /home/nemo en nuestro caso. En cualquier momento que se refiera a un fichero, puede hacerlo en relación a su directorio de trabajo actual, en lugar de especificar la ruta de acceso completa del fichero.

Por tanto, si comienza el nombre de un fichero (como nota1) con un carácter distinto a /, el sistema supone que se está refiriendo al fichero con su posición relativa a su directorio de trabajo. Esto es conocido como ruta de acceso relativa (también PATH relativo). Por otra parte, si comienza el nombre del fichero con /, el sistema interpreta esto como una ruta de acceso completa es decir, la ruta de acceso al fichero completa desde el directorio raíz, /. Esto es conocido como ruta de acceso absoluta (también PATH absoluto).

REFIRIÉNDOSE AL DIRECTORIO HOME

Como vimos anteriormente, podemos referirnos al directorio home usando el carácter de la tilde (~). El carácter ~ es simplemente sustituido por el intérprete de comandos con el nombre del directorio home. El uso de la tilde es simplemente un atajo; no existe ningún directorio llamado ~ es simplemente una ayuda sintáctica proporcionada por el intérprete de comandos. También podemos hacerlo usando la variable de entorno $HOME, la shell sustituirá la variable por su valor que es /home/usuario

EL EDITOR VI, AHORA LLAMADO VIM

vi es un editor de texto. En Linux se debe de conocer lo más básico para poder utilizar **vi** porque está presente (o lo están sus implementaciones más modernas como **vim**), en todos los sistemas **UNIX** (y por lo tanto también en todos los **Linux**). La sintaxis es sencilla, se escribe el comando y a continuación el archivo que se pretende editar (si ése archivo no existe **vi** lo creará).

Modo comando.

Para empezar:

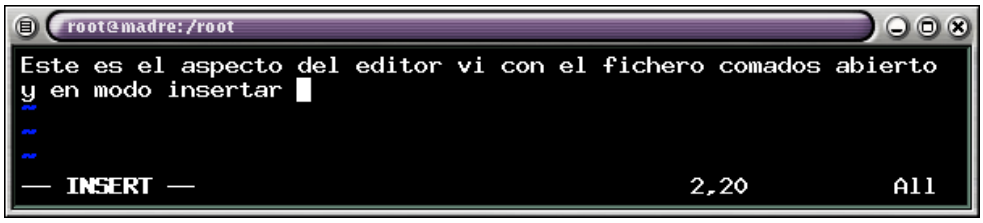
vi [archivo/existente/nuevo]

Una vez que el fichero está abierto **vi** nos muestra su contenido (si lo hay).

vi

Situar el cursor: Casi todas las versiones de **vi** o **vim** soportan ya el uso de las "teclas de dirección" para situarse en el lugar del documento que queremos editar. Pero en caso de que no sea así, podemos utilizar las teclas **h j k l** para mover el cursor hacia la *izquierda, abajo, arriba, derecha* respectivamente. Una vez que tenemos el cursor situado allí donde queremos escribir, debemos pasar al: **Modo inserción**.

Insertar caracteres: Podemos hacerlo de varias formas pero, si pulsamos **i** (letra i), en la parte baja de la pantalla aparecerá la palabra "insert" y podremos escribir en el lugar donde se encuentra el cursor. Si pulsamos **a** (letra a), podremos insertar texto a continuación del cursor.

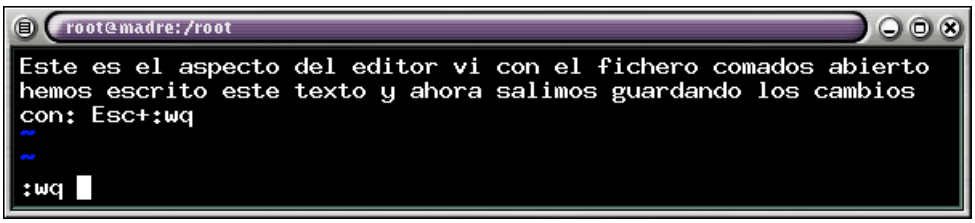


vi

Modo última línea.

Guardar el documento: Tanto para ir guardando el documento mientras editamos como para guardarlo al finalizar, debemos pasar de nuevo al modo última línea tecla **Esc** (escape) y a continuación **:** (dos puntos). Una vez que en la parte inferior izquierda de la pantalla aparecen los dos puntos, para guardar debemos teclear : **w** y a continuación **Intro**.

Salir sin guardar: Si nos hemos equivocado, no queremos que los cambios realizados en el documento tengan efecto o simplemente queremos salir, debemos pasar también al modo última línea (lo hacemos con **Esc + :**) y cuando en la última línea aparezcan los dos puntos "**:"** escribimos "**q**".



vi

guardar y salir: o hacemos en modo última línea (Esc+:) con **wq** .

En ocasiones el editor no nos dejará guardar o salir de un documento que hemos abierto (probablemente porque no teníamos permisos de escritura en ese fichero), si no nos deja guardar los cambios no podemos hacer nada, pero podemos intentar forzarle a hacerlo. Para forzar a **vi** debemos de usar el símbolo imperativo **!** A continuación del comando, de esta forma : **q!**, **w!** o **wq!** .

CREACIÓN DE UN SCRIPT

Conociendo el uso de un editor de texto, la realización de un script de shell es una tarea trivial, si este script lo es también, claro.

Generalmente crearemos scripts para uso personal, como automatización de tareas o ejecución de sentencias largas de ordenes. También es posible que usemos scripts de terceros e incluso que otros leguen a utilizar un script nuestro.

Existe un carácter especial del cual haremos bastante uso: la almohadilla (#). Toda línea que comience por almohadilla será ignorada por el interprete de comandos. Es lo que se conoce como **comentario**.

Normalmente se usa para documentar los scripts o para omitir la ejecución de una línea determinada. (Recuerde que la shell leerá línea a línea el contenido del script, cada línea una orden.)

Además la usaremos para otra cosa más, la primera línea del script se suele empezar así:

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

Es la forma de indicar al interprete de comandos que lo que viene a continuación es un script y como debe usarlo. Se suele indicar sh por ser un interprete que se encuentra en todas las maquinas Linux y Unix.

A partir de aquí el script sigue según nuestras necesidades. Puede que necesite usar variables o simplemente ejecutar dos docenas de comandos. El uso de caracteres especiales le será útil en la creación de potentes scripts.

Aquí unos ejemplos sencillos:

```
#!/bin/bash
read ARCHIVO
/usr/bin/man -t $ARCHIVO >printman.ps
#Este es un script para imprimir en un archivo ps cualquier página del man
```

Todavía quedan un par de pasos para que los scripts sean del todo operativos:

1. Dar al script permiso de ejecución.

2. Poner el script en un directorio que esté incluido en el ***PATH.***

Un script se puede lanzar como argumento de una shell:

[nemo@madre~]$ sh printman

Sí el script anterior se llamase printman, la sentencia anterior lo lanzaría. Esto es igual que:

[nemo@madre~]$ .printman

Fíjese en el punto! Otro ejemplo, asignamos permiso de ejecución, pero el directorio donde se encuentra no está en el PATH:

[nemo@madre~]$./printman

El "./" (punto - barra) indica al shell que el script se encuentra en el directorio actual y lo ejecuta. Este método es perfecto para probar el script o la ejecución esporádica. Lo más correcto para un script de uso habitual es colocar el script en un directorio que esté incluido en el PATH.

Recuerde que los scripts que realicen tareas administrativas pueden ir al directorio ***/sbin***, los generales para todos los usuarios a ***/bin***, los personales se deberían guardar en ***/home/user/bin*** (sustituyendo user por el nombre del usuario en cuestión) y que hay que vigilar bien la asignación de permisos.

ENLACES DE FICHEROS

Los enlaces le permiten dar a un único fichero múltiples nombres. Los ficheros son identificados por el sistema por su numero de inodo

Un inodo es un enlace el cual es el único identificador del fichero para el sistema de ficheros.

Un directorio es una lista de números de inodo con sus correspondientes nombres de fichero. Cada nombre de fichero en un directorio es un enlace a un inodo particular.

ENLACES DUROS (HARD LINKS)

Usando **ls -i,** veremos el numero de inodo para el fichero. La orden **ls -l** muestra el numero de enlaces a un fichero (entre otra información). La orden **ln** es usada para crear enlaces para un fichero

Estos enlaces son conocidos como **enlaces duros** *(hard link*s) porque directamente crean el enlace al inodo. Nótese que **sólo** podemos crear enlaces duros entre ficheros del mismo sistema de ficheros; los enlaces simbólicos (ver más adelante) no tienen esta restricción.

Un fichero es solo definitivamente borrado del sistema cuando no quedan enlaces a el. Usualmente, los ficheros tienen un único enlace, por lo que el uso de **rm** los borra. Pero si el fichero tiene múltiples enlaces, el uso de rm solo borrara un único enlace; para borrar el fichero, deberá borrar todos los enlaces del fichero.

La segunda columna en el listado de la *figura 3*, especifica el numero de enlaces al fichero. Así resulta que un directorio no es más que un fichero que contiene información sobre la dirección del enlace al inodo. También, cada directorio tiene al menos dos enlaces duros en el: «**.**» (un enlace apuntando a si mismo) y «**..**» (un enlace apuntando al directorio padre).

En el directorio raíz (/), el enlace «..» simplemente apunta a /.

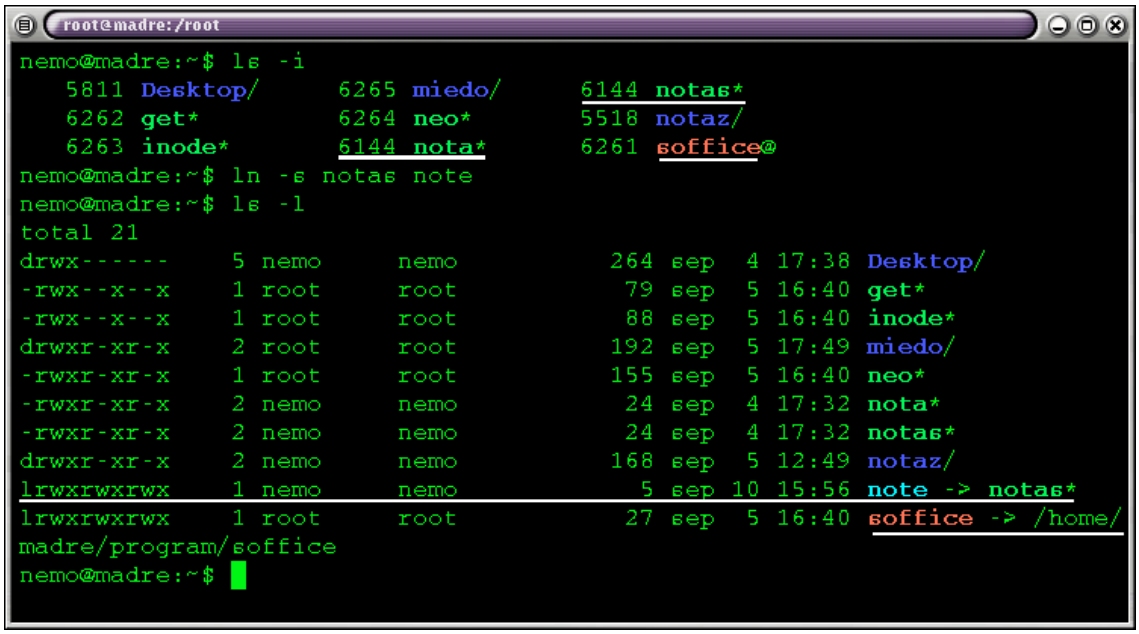


figura 3

ENLACES SIMBÓLICOS O BLANDOS (SOFT LINKS)

Un enlace simbólico permite dar a un fichero el nombre de otro, pero no enlaza el fichero con un inodo.

La orden **ln -s** crea un enlace simbólico a un fichero.

En la *figura 3* vemos que usando la opción **-i** del comando **ls** vemos que el fichero "nota" es un enlace duro de "notas" (tienen el mismo número de inodo), vemos también que señala el enlace blando "soffice" con una **@.** Nosotros creamos otro enlace blando que apunta al fichero "notas".

El enlace se llama "note" y nótese que el la orden **ls -l** nos lo muestra en color azul claro (no disponible en todas las distribuciones de GNU/Linux) y sobre todo con un símbolo **->** que apunta al fichero "real". Los bits de permisos en un enlace simbólico no se usan (siempre aparecen como (rwxrwxrwx). En su lugar, los permisos del enlace simbólico son determinados por los permisos del fichero «apuntado» por el enlace (en nuestro ejemplo, el fichero notas).

Funcionalmente, los enlaces duros y simbólicos son similares, pero hay algunas diferencias. Por una parte, se puede crear un enlace simbólico a un fichero que no está en el mismo dispositivo de almacenamiento; cosa que no es posible (de momento) con enlaces duros. Los enlaces simbólicos son procesados por el núcleo de forma diferente a los duros, lo cual es solo una diferencia técnica, pero a veces importante. Los enlaces simbólicos son de ayuda puesto que identifican al fichero al que apuntan; con enlaces duros no es tan fácil saber que fichero esta enlazado al mismo inodo.

Los enlaces se usan en muchas partes del sistema Linux. Los enlaces simbólicos son especialmente importantes para las imágenes de las librerías compartidas en /lib.

SÚPER BLOQUE

Los números de inodo que identifican a cada fichero son adjudicados por el kernel. Cuando se establecen las particiones y se hace el sistema de ficheros (bien sea durante el proceso de instalación o posteriormente), el núcleo escribe información (tamaño, **FS**, situación) sobre estas particiones en una tabla que está al principio del **HD** junto al **MBR**. Cada partición dispone de un **SB** (súper bloque), al que el kernel adjudica un rango determinado de números de inodo. En núcleos antiguos estos rangos eran "limitados", en núcleos modernos son proporcionales al tamaño y "posible uso" de la partición y no tenemos noticia de que se hayan agotado. Cuando el usuario o el propio sistema operativo crea un fichero o un directorio, el núcleo mira el SB del FS donde se está creando y adjudica uno de esos números de inodo a ese fichero. Este es el motivo por el que no se puede hacer enlaces duros entre ficheros que están en particiones o sistemas de ficheros diferentes.

Los modernos FS compatibles con Linux como **ReiserFS**, además de las conocidas características de journaling, aprovechamiento del espacio en disco y mejoramiento de la fragmentación interna y externa, cuentan con "asignación dinámica de espacio para los inodos"" . Es decir, no se reservan bloques específicos para inodos, así que no es necesario preveer los inodos que necesitaremos.

GESTIÓN DE USUARIOS Y CUENTAS DE ROOT

La figura del administrador del sistema en Linux tiene nombre propio: **root**.

LA CUENTA ROOT

Es la cuenta del administrador y no hay limitaciones para este usuario. Esto es lo primero y más veces repetido: **¡No use la cuenta root habitualmente!**. Solo se debe emplear para realizar tareas administrativas. **Si se comete un error siendo root, el sistema no le detendrá.**

La formula habitual, si se requieren los poderes del root, es utilizar el comando «***su***». Este comando ejecuta una shell con identificadores de usuario y grupo distintos.

Lo que quiere decir que permite a un usuario convertirse temporalmente en otro usuario. Si no se especifica ningún nombre de usuario, por defecto se usa root. La *shell* a ejecutar se toma de la entrada correspondiente al usuario en el fichero de *contraseñas*, o***_/bin/sh*** si no está especificada en dicho fichero. La ejecución de su solicitará el contraseña del usuario, menos que se ejecute por el root.

El administrador será también el encargado de la realización de cambios en el sistema. Cualquier cambio deberá ser cuidadosamente planificado y probado en un entorno controlado antes de implantarlo en producción.

Es imposible evaluar todas las posibles variables y condiciones que se generarán al llegar los usuarios al día siguiente de realizar ese cambio tan importante. Pero inténtelo, la responsabilidad es suya.

El administrador debe ser el que provee las soluciones, si su actividad genera problemas, tenga previsto como restaurar una condición estable en el mínimo lapso de tiempo. No solo debe pensar como implementar el nuevo cambio, además tenga previsto un plan de recuperación de emergencia, por si algo sale mal.

También es buena idea plantear gradualmente las mejoras, es más fácil volver atrás si los cambios son menores que cuando se cambian de golpe gran número de cosas. No sabrá que falló ni porque si hizo todos los cambios a la vez. Y no digamos si afectan a gran parte del equipo. Podrá diagnosticar un problema si sólo tocó un par de cosas y comprobó el funcionamiento antes de seguir adelante.

Cada cambio debe ser anunciado con suficiente antelación cuando afecte a los usuarios, de manera que sepan como comportarse ante la novedad. El mejor sistema, funcionando correctamente puede ocasionar una pequeña revuelta si los usuarios ven afectado su trabajo, aunque solo hablemos de pequeños cambios de forma. Infórmeles con antelación de lo que va a ocurrir y documente, si es necesario, las nociones sobre el manejo (o lo que sea) que necesiten.

DEB Y RPM, GESTIÓN DE PAQUETES

Los sistemas de empaquetado de software, se están convirtiendo en el método preferido por los cada día más numerosos usuarios de **GNU/Linux**, para la gestión de sus programas, aplicaciones y bibliotecas . Los formatos **deb (Debian Package)** y **rpm (RedHat Package Manager)** proporcionan al usuario final una serie de facilidades que hacen más sencillo el mantenimiento del sistema. **En ESware Linux 365** se pueden usar los dos formatos (aunque lo más aconsejable es instalar **paquetes.deb**) Para evitar problemas con las dependencias, siempre que necesitemos instalar un **paquete.rpm** o un **paquete.tgz**, podemos convertirlo en **paquete.deb** (como explicamos unas líneas más abajo), utilizando la aplicación **alien** . Son muchas las ventajas que **deb** y **rpm** ofrecen sobre otros modelos de empaquetado como **tar.gz,** quizá más adecuado para programadores, desarrolladores y expertos que necesiten trabajar con el código fuente de los programas . Tanto **deb** como **rpm** mantienen una base de datos con mucha información sobre los paquetes instalados y de los archivos que contienen, lo que permite realizar consultas y en algunos casos verificaciones de gran utilidad .

Esto propicia además que la desinstalación sea "muy limpia", cuando un paquete se desinstala no queda nada de el en el sistema, a excepción de algunos archivos de configuración que en algunas ocasiones deberemos "borrar" nosotros mismos, en todo caso siempre seremos alertados sobre la existencia de esos ficheros . Los dos formatos respetan (en la mayoría de las ocasiones), los archivos de configuración al actualizar software, de manera que no sea necesario realizar ajustes específicos que ya estuvieran definidos. Cuando no es posible respetar estos ficheros se realizan copias de seguridad de los mismos . Los dos sistemas de paquetes se distribuyen bajo los términos de la **GPL** (General Public License) .

APT, DPKG, RPM, DSELECT, APTITUDE, CONSOLE-APT, KPACKAGE, GNORPM, GNOME-APT, STORMPKG , XRPM, PURP Y GLINT

Los dos formatos, **deb** y **rpm**, disponen de aplicaciones que nos permiten la gestión desde la línea de comandos : **apt, dpkg, rpm, dselect, aptitude y console-apt** . También existen diferentes front-ends que facilitan el manejo de paquetes en el entorno gráfico : **kpackage, gnorpm, gnome-apt, stormkpg, xrpm, purp y glint** .

KDE ofrece a sus usuarios el la aplicación **kpackage** que gestiona paquetes **deb y rpm**, **GNOME** cuenta con **gnorpm**, que como su nombre indica se ocupa del manejo de paquetes rpm y **gnome-apt** para paquetes **deb**. La aplicación gráfica **stormpkg** es también un excelente front-end para la gestión de paquetes **deb** . Las aplicaciones **xrpm, purp y glint**, también son capaces de pasarle a **rpm** los parámetros necesarios para la gestión de paquetes.

Dado que el manejo de estas aplicaciones gráficas es suficientemente intuitiva, dedicaremos el espacio y el tiempo a escribir acerca de **apt**, dpkg y **rpm** (dselect a pesar de su antigüedad sigue presente en algunas distribuciones, pero dada su "dificultad" está cayendo en desuso, lo mismo ocurre con xrpm, purp y glint) .

APT (APT-GET)

Comenzaremos con la más "moderna" , **apt** viene de : Advanced Package Tool, en una herramienta para la gestión de paquetes **deb**, sin duda la más avanzada actualmente, sobre todo por su flexibilidad y potencia para entornos de red . Como interface en la línea de comandos nos ofrece **apt-get** .

Para poder usar **apt** (además de tener el paquete **apt** instalado y una conexión a **Internet**), debemos de editar el archivo **sources.list** que está en : /etc/apt/sources.list . En este fichero debemos poner las direcciones de los servidores de donde **apt-get** debe recoger los paquetes para su instalación o actualización en nuestro sistema. Así cada vez que ejecutemos la orden : **apt-get update**, **apt** actualizará la lista de paquetes disponibles en los lugares indicados en el sources.list . Una vez definido este fichero ya podemos trabajar con **apt-get** que, además de todas las operaciones que se definen a continuación, **resuelve de forma automática las dependencias** .

COMANDOS BÁSICOS APT-GET

apt-get update actualiza la lista de paquetes disponibles en los lugares que figuran en /etc/apt/sources.list

apt-get install <paquete> instala y actualiza paquetes. Si el que se pretende instalar es más antiguo nos avisará y tendremos que decidir que hacer .

apt-get remove <paquete> desinstala paquetes

apt-get source <paquete> "baja" las fuentes del paquete

apt-cache search <cadena de caracteres> busca paquetes

apt-cache show <paquete.deb> nos ofrece información sobre el paquete

apt-get upgrade actualiza todos los paquetes "desfasados" (versiones antiguas)

apt-get dist-upgrade actualiza todo . **Instala la siguiente versión de la distribución**

apt-get install [task-paquete.deb] instala además otros paquetes de aplicaciones que "van" con el nombrado aunque no sea necesario instalarlos por dependencias .

DPKG

Se trata de otra aplicación para la gestión de paquetes **deb** . **dpkg** no resuelve automáticamente las dependencias. Si le damos una orden para instalar un paquete, comenzará a instalarlo y si durante el proceso se encuentra con que el paquete "necesita" de otros programas o bibliotecas, **detiene la instalación y nos informa de los paquetes que debemos de instalar antes del nombrado** . Por ello, algunos usuarios prefieren usar **dselect** (utilidad que maneja **dpkg**), porque muestra las dependencias (y sugiere además otros paquetes no estrictamente necesarios), del paquete en cuanto lo seleccionamos, dándonos la posibilidad de elegir lo que queremos hacer en en ese instante .

COMANDOS BÁSICOS DPKG

dpkg -l ofrece una lista de todos los paquetes instalados . La salida del comando puede entubarse al comando **grep**, que podemos utilizar para "buscar paquetes" .

dpkg -s <paquete> nos ofrece información sobre paquetes instalados

dpkg -i <paquete.deb> instala el paquete que le pasamos como argumento

dpkg -r <paquete.deb> desinstala el paquete (sin borrar los ficheros de configuración)

dpkg --purge <paquete.deb> desinstala el paquete (borrando también los archivos de configuración)

dpkg -L <paquete> lista el contenido paquete si esta instalado

dpkg -c <paquete.deb> muestra todos los ficheros que contiene un paquete no instalado

dpkg --info <paquete.deb> ofrece información del paquete no instalado

dpkg -S <fichero> nos dice en que paquetes está el fichero que le pasamos como argumento

En la web http://packages.debian.org los usuarios de **deb** disponen de un buscador de paquetes, con el que pueden localizar cualquier versión de un paquete a partir de su nombre o, incluso del nombre de un fichero perteneciente a ese paquete

RPM

rpm es una aplicación para la gestión de paquetes en formato RPM (Red Hat Package Manager) . Los paquetes se nombran utilizando todos los datos posibles, por ejemplo : **eswason-0.1-3.i386.rpm** primero el nombre del paquete (**eswason**), la versión(**-0.1**), desarrollo (**-3**), y arquitectura (**i386**) y formato (**rpm**). Debemos asegurarnos de elegir la arquitectura adecuada para nuestro sistema ! . **rpm tampoco resuelve las dependencias de forma automática**, pero si le ordenamos instalar un paquete y éste tiene dependencias con otros paquetes o bibliotecas que están en otros paquetes, se negará a instalarlo y nos informará acerca de los paquetes que tenemos que instalar antes del indicado . Podemos "forzar" a **rpm** para que instale de todas formas con la opción **--nodeps**, pero lo opción más aconsejable es respetar las dependencias de cada paquete .

COMANDOS BÁSICOS RPM

rpm -qa (imprime una lista de todos los paquetes instalados (la extrae de : **/var/lib/rpm/packages.rpm**) . La salida del comando acepta una un pipe " | " a la entrada de **grep**, que podemos utilizar para "buscar paquetes" .

rpm -ivh <paquete.rpm> instala el paquete que se le pasa como argumento

rpm -e <paquete.rpm> desinstala el paquete señalado

rpm -Uvh <paquete.rpm> actualiza, desinstalando la versión antigua del paquete

rpm -test <paquete.rpm> hace una prueba de instalación (podemos ver lo que ocurriría si instalásemos)

rpm -ivh --replacepkgs <paquete.rpm> "actualiza" el paquete aunque el que queremos instalar sea más antiguo que el que está instalado . Se fuerza a **rpm** a reemplazar el paquete existente .

rpm -ivh --replacefiles <paquete.rpm> instala el paquete y reemplaza los archivos ya instalados por otro paquete . (Usaremos la opción **--replacefiles** si **rpm** nos dará un error al instalar un paquete que contiene uno o más archivos que ya han sido instalados en el sistema por otro paquete) .

rpm -ivh --nodeps <paquete.rpm> instala el paquete obviando las dependencias (esta opción puede comprometer la estabilidad y el funcionamiento normal del programa instalado) .

rpm -qi <paquete.rpm> nos da una descripción del contenido del paquete

rpm -qf <fichero> imprimirá en pantalla el paquete al que pertenece el fichero que le damos como argumento

rpm -V <paquete> realiza una profunda verificación del paquete instalado y muestra cualquier modificación que haya sufrido desde la fecha de su instalación . Si no imprime nada en pantalla, es que el paquete no ha sido alterado. Puede usarse para verificar todos los paquetes escribiendo : **rpm -V `rpm -qa`** (esta operación puede durar demasiado en una máquina antigua) .

En la web http://rpmfind.net, los usuarios de **rpm** disponen de un buscador de paquetes, con el que pueden localizar cualquier versión de un paquete a partir de su nombre o, incluso del nombre de un fichero perteneciente a ese paquete .

ALIEN (CONVERTIR PAQUETES DEB, RPM Y TGZ)

alien es una aplicación "capaz" de convertir paquetes **dev** a **rpm, rpm** a **dev, dev** a **tgz, rpm** a **tgz y tgz** a cualquiera de los anteriores e incluso instalarlos. Alien realiza la conversión descomprimiendo el paquete original y generando el nuevo paquete y además nos informa del lugar donde "pondrá" el paquete nuevo. Este lugar no es siempre el mismo y cada distribución tiene sus preferencias al respecto .

COMANDOS BÁSICOS DE ALIEN

alien --to-deb <paquete.rpm o **paquete.tgz>** genera un paquete **deb**

alien -d <paquete.rpm o **paquete.tgz>** genera un paquete **deb**

alien --to-rpm <paquete.deb o **paquete.tgz>** genera un paquete **rpm**

alien -r <paquete.deb o **paquete.tgz>** genera un paquete **rpm**

alien --to-tgz <paquete.deb o **paquete.rpm>** genera un paquete **tgz**

alien -t <paquete.rpm o paquete.deb> genera un paquete **tgz**

alien -i <paquete.rpm> convierte el paquete a **deb** y lo instala

alien -i [paquete.tgz] convierte el paquete a **deb** y lo instala

INSTALACIÓN DE APLICACIONES EN FORMATO .TAR.GZ

COMANDOS BÁSICOS

./configure

make

make install

No suele haber dificultades a la hora de instalar programas empaquetados en .tar.gz. En ocasiones podemos encontrarnos con que, al desempaquetar-descomprimir la aplicación, en el directorio resultante aparecen binarios. Esto quiere decir que ya había sido compilada antes de tarearla . En este caso el programa ya estará instalado y sólo habrá que lanzar el ejecutable que estará entre los archivos del directorio mencionado.

Lo habitual es que al desempaquetar nos encontremos con el código fuente del programa (en C). En este caso son necesarios los comandos mencionados anteriormente y los problemas pueden surgir si no se siguen los pasos adecuadamente. Veamos un ejemplo.

Hemos bajado de Internet un un archivo que contiene la versión española de las páginas del **man** en español: **man-pages-es-extra-0.8a.tar.gz** y **man-pages-es-1.28.tar.gz**. Vamos a instalarlas.

En primer lugar nos situamos en el directorio que contiene estos dos archivos, a continuación "abrimos" uno de ellos con el comando:

tar -xzvf **man-pages-es-1.28.tar.gz**

El comando tar desempaquetará, descomprimirá y alojará en un directorio (que crea a tal efecto), el contenido del paquete: **man-pages-es-1.28.tar.gz**. El directorio se llamará: **man-pages-es-1.28/**.

A continuación debemos entrar en ese directorio con el comando:

cd /man-pages-es-1.28

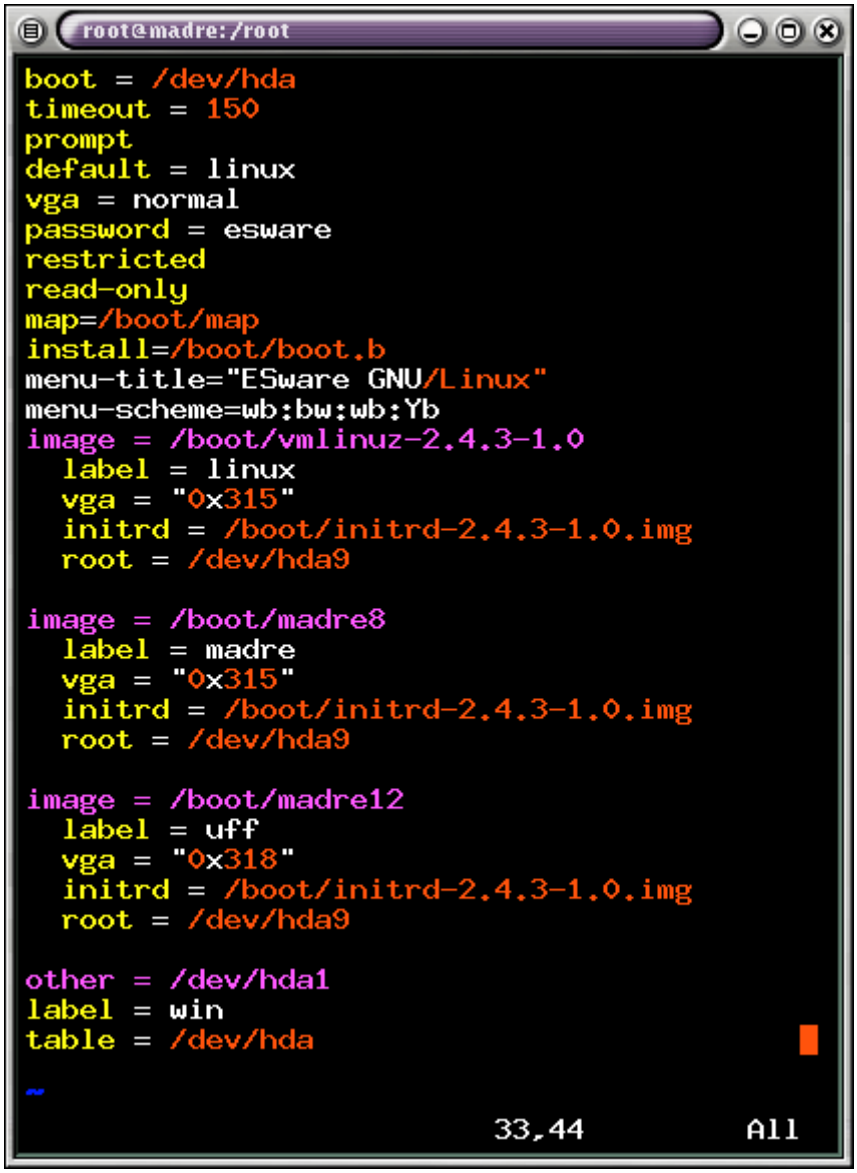
Y una vez en el, listarlo (con ***ls***) y leer el archivo: README o LEEME (que para eso están). El creador del paquete incluirá en este archivo toda la información necesaria para su instalación (en ocasiones es necesario realizar "otras" operaciones, además de ejecutar los comandos que mencionábamos al principio.

Continuamos ejecutando **./configure** (ejecuta el script configure en el directorio actual, este script revisa nuestro sistema "mirando" todas las opciones de configuración que pueden afectar al paquete que pretendemos instalar. En caso de encontrar alguna dificultad nos avisará), después **make** (esta es una orden de compilación "hacer" los binarios) y a continuación **make install** (instala los binarios que ha hecho en el paso anterior, teniendo en cuenta las opciones de configure). La aplicación quedará instalada (normalmente en /usr/bin). Si el archivo README (LEEME), indicaba alguna operación posterior debemos realizarla ahora.

En este caso (**man-pages-es-1.28/**.), es necesario revisar y en su caso modificar los archivos: /home/usuario/.bash_profile y /etc/man.config.

GESTIÓN DE ARRANQUE LILO

Linux Loader tiene un fichero binario (ejecutable) que es el encargado de escribir en el MBR la información de la configuración que deseemos. Este fichero es /sbin/lilo y la información que debe grabar en el MBR la lee del fichero de configuración /etc/lilo.conf.



Fichero de configuración de lilo, /etc/lilo.conf

Se distinguen dos secciones en el fichero /etc/lilo.conf, una es para especificaciones "globales" (las seis primeras líneas) y en la otra se definen las "imágenes". Las "imágenes" son configuraciones especificas para cada uno de los sistemas que podemos arrancar. Desde luego, las opciones que se pongan en la sección "global" afectarán a todas las imágenes. El fichero lilo.conf puede tener hasta 16 imágenes.

La forma de modificar la configuración de LILO es la siguiente:

1. Realizar los cambios en el fichero /etc/lilo.conf
2. Ejecutar /sbin/lilo para que estos cambios se escriban en el MBR. Sin este paso los cambios que haya realizado ¡no tendrán ningún efecto!

- **Opciones de lilo.conf**

·**Opciones Globales**

Veamos las opciones más interesantes que podemos utilizar en el fichero /etc/lilo.conf:

boot=dispositivo-de-arranque

Indica el nombre del dispositivo que contiene el sector de arranque.

compact

Intenta agrupar operaciones de lectura en sectores adyacentes en una sola operación. Esto reduce drásticamente el tiempo de carga y hace que el mapa sea mas pequeño. Se recomienda usar `compact' cuando se arranca de un disco flexible.

default=nombre

Indicaremos aquí la imagen especificada para que arranque por omisión. Si se omite esta entrada, se utilizará la imagen que aparezca en primer lugar en el fichero /etc/lilo.conf.

delay=décimas-seg

Tiempo (en décimas de segundo) que Lilo esperará antes de lanzar la imagen por defecto. Si se omite o si se pone a 0 no habrá tiempo de espera.

timeout=décimas-seg

Si no se aprieta ninguna tecla en el tiempo (en décimas de segundo) especificado, se lanza automáticamente la primera imagen. Es un límite de tiempo para entradas por teclado.

map=fichero-mapa

Especifica la ubicación del fichero mapa.

message=fichero-mensajes

Puede hacer que aparezca un mensaje antes del prompt de Lilo. Aquí especificaría un fichero con el texto del mensaje. No puede exceder de 65536 bytes.

·Opciones para la Imágenes

Cada imagen empieza con la línea:

image=nombre-de-ruta

en el caso de que la imagen de arranque pertenezca a un núcleo Linux, o con la línea:

other=nombre-de-ruta

para lanzar cualquier otro sistema. En este caso se debe indicar una de estas opciones:

table=dispositivo

Indica el dispositivo que contiene la tabla de particiones. Lilo no va a enviar información de partición al sistema operativo si se omite esta variable.

loader=cargador-cadena

Indica que debe emplearse un cargador en cadena.

unsafe

Indica que lilo no debe acceder al sector de arranque cuando cree el mapa. `unsafe' y `table' son incompatibles.

Las opciones comunes para imágenes Linux y de otros sistemas son:

label=nombre

Nombre para identificar la imagen. Es el que deberemos indicar en el prompt de LILO.

alias=nombre

Define un segundo nombre (o alias) para la misma imagen.

password=contraseña

Pide contraseña al intentar lanzar la imagen.

restricted

Al poner esta opción, sólo pedirá contraseña si se especifican parámetros en el prompt de LILO.

·Opciones del núcleo

Para imágenes de núcleos Linux, disponemos además de otras opciones.

read-only

Monta el sistema de ficheros raíz en modo "solo lectura", hasta después de hacer la comprobación que realiza en el arranque. Pasado este punto se re-monta ya en modo "lectura escritura"

root=dispositivo-raíz

Especifica el dispositivo que debe ser montado como sistema de ficheros raíz.

vga=modo

Esta opción especifica el modo de texto VGA que se usará al arrancar el sistema. Son válidos los siguientes valores:

normal: modo de texto 80×25.

extended:modo de texto 80×50.

ask: pregunta al usuario

<número>: modo de texto correspondiente al número.

Para obtener una lista de modos disponibles, arranque con 'vga=ask' y pulse [Intro].

El ejecutable /sbin/lilo, tiene ademas numerosas opciones:

-v Muestra más información durante la ejecución.

-q Da una lista de los ficheros en el fichero mapa (/boot/map,) que contiene el nombre y lugar del (de los) núcleo(s) a arrancar. Esta opción muestra los nombres en dicho fichero.

-m fichero-mapa
Emplea el mapa especificado, en lugar del predeterminado.

-C fichero-configuración
Se usa para especificar otro fichero de configuración distinto de /etc/lilo.conf.

-t Test. Realiza una prueba sin escribir realmente un nuevo MBR ni el fichero-mapa. Puede usarse en combinación con -v para ver lo que lilo está a punto de hacer.

-s backup_boot_sector

Al escribir el sector de arranque, lilo guarda el contenido previo en un fichero, por omisión /boot/boot.XXXX (donde XXXX depende del dispositivo. Esta opción especifica un fichero alternativo para guardar el sector de arranque. (Usado con la opción -u, especifica desde dónde restaurar el sector de arranque.)

-u nombre-dispositivo

Desinstala lilo, simplemente copiando el sector de arranque que previamente salvó.

Si usted está planeando hacer que Linux coexista con otro sistema operativo, verá las ventajas que aporta el uso de Lilo, pero tenga en cuenta que algunos sistemas operativos escriben directamente (y sin preguntar) en el MBR para controlar el arranque. Si va a usar uno de estos, (por ejemplo Windows 95/98), recuerde que instalarlo después de tener instalado Linux, le hará perder el control del arranque con Lilo ya que será sobrescrito por esta instalación tan amable.

Esto lo evita instalando Linux en último lugar ya que es mucho más respetuoso y coexiste sin ningún tipo de problema con otros, o en el caso de haber perdido ya el contenido del MBR, sencillamente arranque con el disco de inicio (¿lo hizo durante la instalación, verdad?), y cuando entre en el sistema como root, vuelva a ejecutar /sbin/lilo. Asunto resuelto.

GESTIÓN DE DISPOSITIVOS FSTAB

Los sistemas de ficheros se pueden adjuntar en Linux a los directorios del sistema (*montaje de un sistema*). Para hacer esta operación hace falta que seamos el superusuario, que exista el lugar de montaje y que el recurso exista en el directorio en el sudirectorio /dev.

Ademas hay que tener en cuenta si se quiere adjuntar en el archivo de montaje (***/etc/fstab***), si se puede montar con el automontador, si será de forma temporal o si se quiere montar desde la linea de comandos con mount .

Si lo queremos montar desde la linea de comandos:

mount dispositivo punto_de_montaje

A esta llamada se le pueden pasan un numero de parámetros tales como:

-t tipo: Con lo que se especifica el sistema de archivos que se esta montando.

-*w:* El sistema se montara con permiso de lectura y escritura.

-*r:* El sistema se montara solo con los permisos de lectura.

Teniendo en cuenta esto, los tipos de archivos que se pueden utilizar son ***ext2*** (el modo de archivos locales con nombres largos), ***msdos*** (Sistemas de archivos del sistema operativo MS-DOS), ***iso9660***(Sistema de archivos de los CDROM), ***nfs*** (Para montar sistemas de archivos remotos), etc..

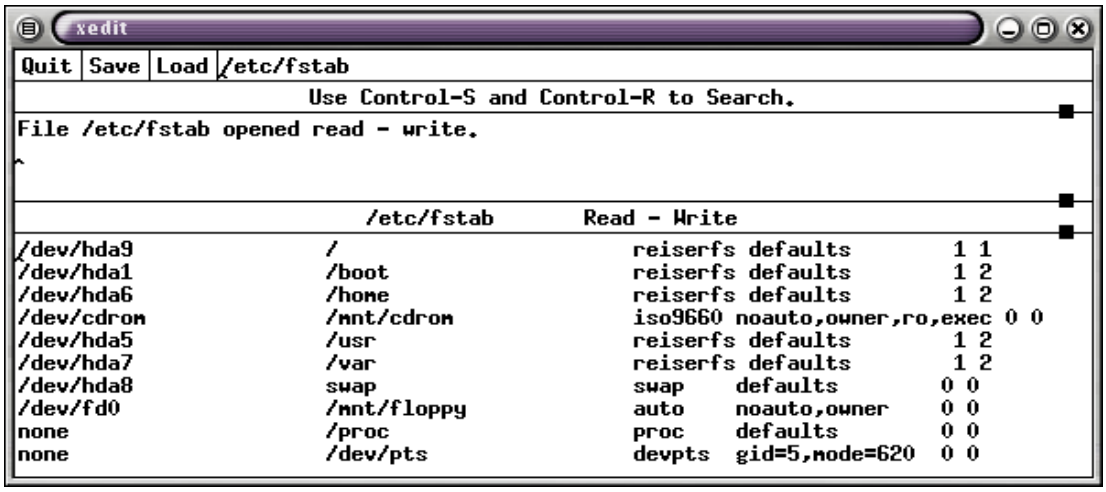
mount -t iso9660 /dev/cdrom /mnt/cdrom

Para desmontar tenemos el comando ***umount***:

umount /mnt/cdrom

Otra forma para montar las unidades es utilizar el archivo ***fstab***. Con esto conseguimos que en el arranque del sistema se monten directamente todos los sistemas de archivos que tengamos.

En el siguiente ejemplo podemos ver como se pueden montar varios tipos de sistemas:



/etc/fstab

Aquí hemos visto como se monta el sistema de inicio, la partición de arranque, el cdrom, la partición swap, la unidad de disco, el sistema de archivos proc y pts y una unidad nfs.

En la tercera columna estará situado el sistema de archivos que se esta montando.

En la cuarta columna de este archivo se van a introducir las opciones que se deseen al montar los archivos.

En la quinta columna se indica que sistemas de archivos necesitan ser volcados. En caso de ser cero no se volcaran.

El sexto campo indica el orden de chequeo al arrancar el sistema.

SISTEMA DE ARCHIVOS /PROC

El sistema de archivos /proc es algo específico de Linux. Es un sistema de archivos virtual, y como tal, no ocupa lugar en su disco. Es una forma muy conveniente de obtener información sobre el sistema, ya que la mayoría de los archivos de este directorio son legibles (bueno, con un poco de práctica). De hecho, muchos programas obtienen información de los archivos de /proc, le dan formato a su manera y luego la muestran.

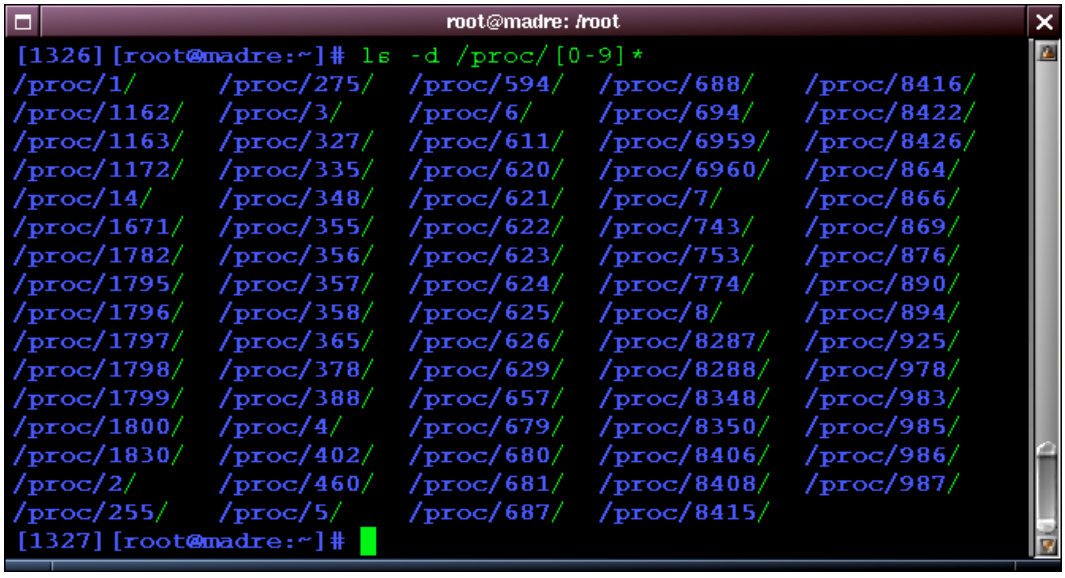
Este es el caso de todos los programas que muestran información sobre los procesos. /proc también es una buena fuente de información sobre su hardware, y similarmente, unos cuantos programas solo son las interfaces de la información contenida en /proc.

También hay un sub-directorio especial, /proc/sys. Este permite cambiar algunos parámetros del núcleo en tiempo real o consultarlos.

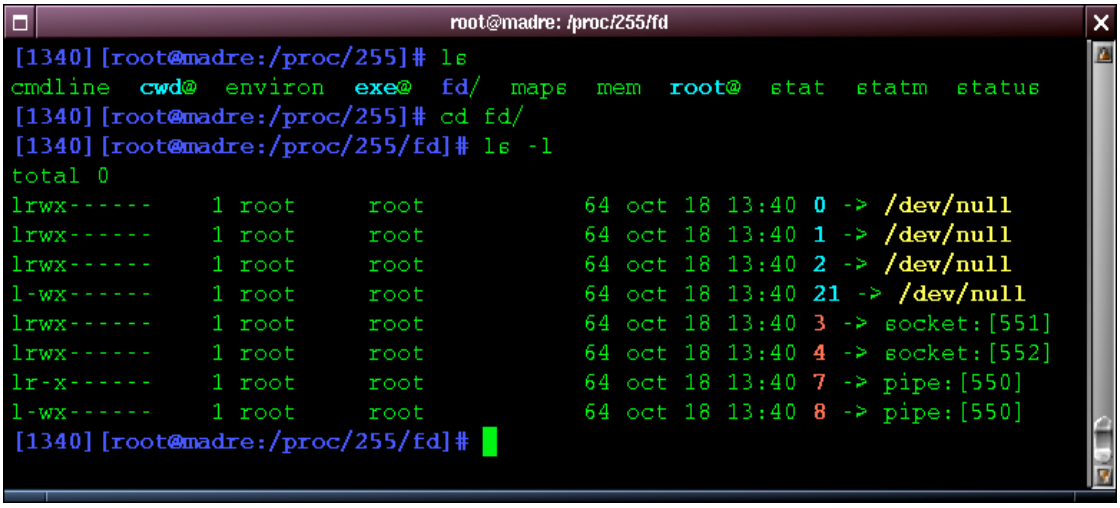
· **Información sobre los procesos**

Si Ud. lista el contenido del directorio /proc, verá muchos directorios cuyo nombre es un número. Estos son los directorios que contienen información sobre todos los procesos funcionando en el sistema en un instante dado:

/proc



Note que como usuario no privilegiado, Ud. (lógicamente) solo puede mostrar la información relacionada con sus propios procesos, pero no con los de los otros usuarios. Entonces, seamos root y veamos que información está disponible del proceso 255 y en su subdirectorio /fd:

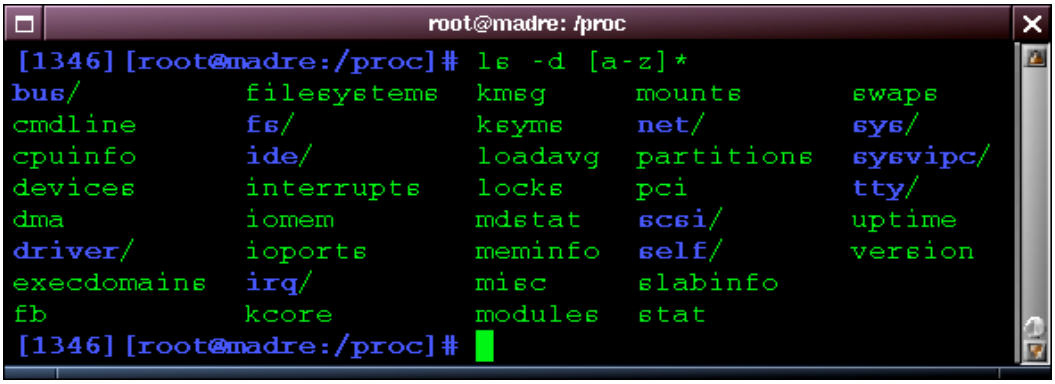


/proc

De hecho, esta es la lista de los descriptores de archivo abiertos por el proceso. Cada descriptor abierto está materializado por un vínculo simbólico cuyo nombre es el número del descriptor, y que apunta al archivo abierto por este descriptor[1]. Ud. también puede notar los permisos sobre los vínculos simbólicos:

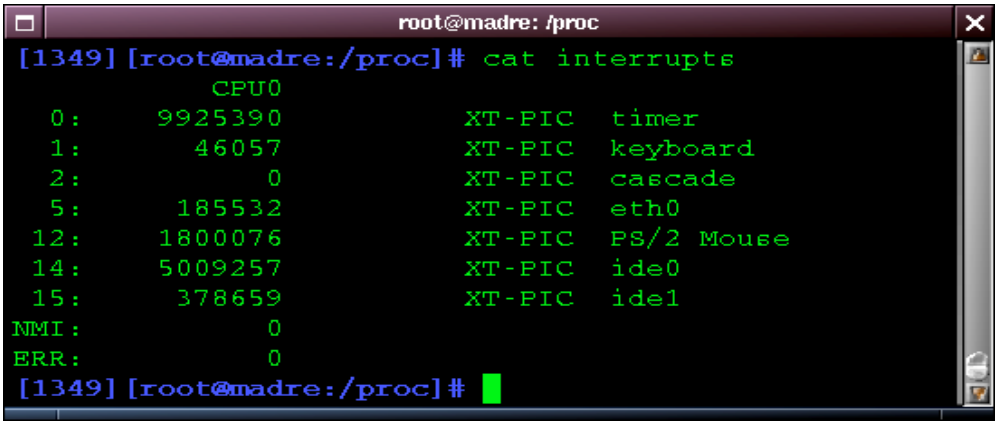
este es el único lugar donde los derechos tienen sentido, ya que representan los permisos con los cuales se abrió el archivo correspondiente al descriptor.

Aparte de los directorios asociados a los diferentes procesos, /proc también contiene una miríada de información sobre el hardware presente en su máquina. Un listado de los archivos del directorio /proc da lo siguiente:

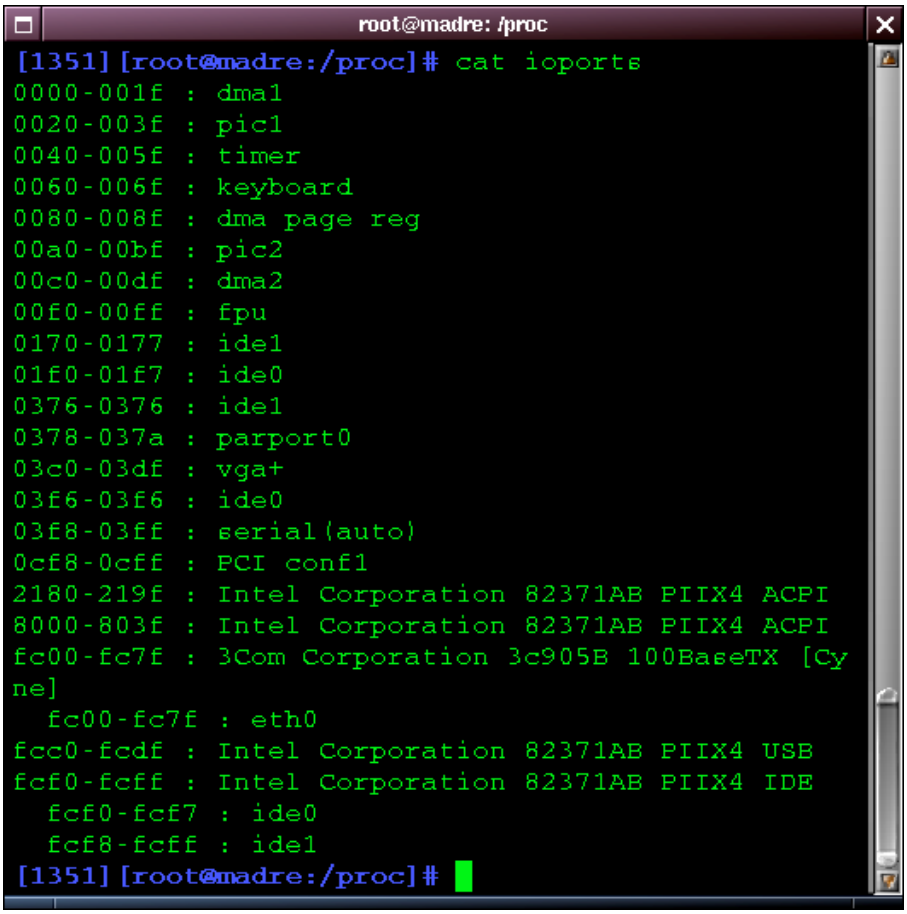


/proc

Por ejemplo, si observamos el contenido de **/proc/interrupts**, podemos ver la lista de las interrupciones que el sistema está usando actualmente, junto con el periférico que las está ocupando. Similarmente, **ioports** contiene la lista de los rangos de direcciones de entrada/salida ocupados en este momento, y finalmente, **dma** hace lo mismo para los canales DMA. Por lo tanto, si Ud. desea solucionar un conflicto, observe el contenido de estos tres archivos:



interrupts



ioports

El sistema de ficheros /proc actúa como una interfaz para las estructuras internas de datos del núcleo. Puede usarse para obtener información sobre el sistema y para cambiar ciertos parámetros del núcleo mientras se ejecuta. Contiene (entre otras cosas) un subdirectorio por cada proceso ejecutándose en el sistema que se llama como el PID del proceso. El enlace self apunta al proceso que esté leyendo el sistema de ficheros.

- Subdirectorios específicos de cada proceso.

Cada subdirectorio de proceso contiene los elementos listados en la tabla 1.1.

cmdline	Argumentos de la línea de comandos.
environ	Valores de las variables de entorno.
fd	Directorio, que contiene todos los descriptores de ficheros.
mem	Memoria retenida por este proceso.
stat	Estado del proceso.
status	Estado del proceso en forma humanamente legible.
cwd	Enlace al directorio actual de trabajo.
exe	Enlace al ejecutable de este proceso.
maps	Mapas de memoria.
root	Enlace al directorio raíz de este proceso.
statm	Información del estado de la memoria del proceso.

Tabla 1.1: Elementos específicos de cada proceso en /proc.

Por ejemplo, para obtener la información de estado de un proceso, todo lo que tiene que hacer es leer el fichero /proc/PID/status:

```
> cat /proc/self/status
Name:   cat
State:  R (running)
Pid:    5633
PPid:   5609
Uid:    501     501     501     501
Gid:    100     100     100     100
Groups: 100 16
VmSize:      804 kB
VmLck:         0 kB
VmRSS:       344 kB
VmData:       68 kB
VmStk:        20 kB
VmExe:        12 kB
```

VmLib: 660 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 00000000fffffeff
CapPrm: 0000000000000000
CapEff: 0000000000000000

Esto le muestra casi la misma información que obtendría si lo viese con el comando ps. De echo, ps usa el sistema de ficheros /proc para obtener la información que muestra.

El fichero statm contiene información más detallada sobre el uso de memoria del proceso. Contiene siete valores que significan lo siguiente:

Size	tamaño total del programa.
Resident	size of in memory portions.
Shared	número de páginas que están compartidas.
Trs	número de páginas que son 'código'.
Drs	número de páginas de datos/pila.
Lrs	número de páginas de biblioteca (library).
Dt	número de páginas marcadas como sucias.

La relación texto/datos/biblioteca sólo se puede aproximar por heurística.

- **Datos del núcleo.**

De manera similar a las entradas para procesos, hay ficheros que dan información sobre el úcleo que está ejecutándose. Los ficheros necesarios para obtener dicha información se hallan en /proc están listados en la tabla 1.2. No todos estos ficheros estarán presentes en su sistema. Depende de la configuración del núcleo y de los módulos que haya cargados, el que estos ficheros existan o no.

apm	Información de la gestión avanzada de energía.
Cmdline	Línea de comandos del núcleo.
Cpuinfo	Información sobre la CPU.
Devices	Dispositivos disponibles (de bloques y de caracteres).
Dma	Canales DMA usados.
Filesystems	Sistemas de ficheros soportados.
Interrupts	Uso de interrupciones.
Ioports	Uso de puertos de E/S (I/O ports).
Kcore	Imagen del centro del núcleo. ¿Kernel core image?
Kmsg	Mensajes del núcleo.
Ksyms	Tabla de símbolos del núcleo.
Loadavg	Carga media.
Locks	Bloqueos del núcleo.
Meminfo	Información de memoria.

Misc	Miscelánea.
Modules	Lista de módulos cargados.
Mounts	Sistemas de ficheros montados.
Partitions	Tabla de particiones que el sistema conoce.
Rtc	Reloj en tiempo real.
Slabinfo	Información de la slab pool
Stat	Estadísticas globales. Overall statistics?
Swaps	Utilización del espacio de intercambio.
Uptime	Tiempo que lleva encendido el sistema.
Version	Versión del núcleo.

Tabla 1.2: Información del núcleo en /proc.

Puede, por ejemplo, revisar qué interrupciones estén actualmente en uso y para qué están siendo usadas mirando en el fichero /proc/interrupts:

```
> cat /proc/interrupts
           CPU0
  0:    8728810        XT-PIC  timer
  1:        895        XT-PIC  keyboard
  2:          0        XT-PIC  cascade
  3:     531695        XT-PIC  aha152x
  4:    2014133        XT-PIC  serial
  5:      44401        XT-PIC  pcnet_cs
  8:          2        XT-PIC  rtc
 11:          8        XT-PIC  i82365
 12:     182918        XT-PIC  PS/2 Mouse
 13:          1        XT-PIC  fpu
 14:    1232265        XT-PIC  ide0
 15:          7        XT-PIC  ide1
NMI:          0
```

Hay tres directorios más de importancia en /proc: net, scsi y sys. La regla de oro es que los contenidos, o incluso la existencia misma de estos directorios depende de la configuración de su núcleo. Si no está activado el soporte scsi, el directorio scsi puede no existir. Lo mismo es cierto con net, que sólo está ahí cuando el soporte de red está presente en el núcleo actual.

- Dispositivos IDE en /proc/ide.

Este directorio contiene información sobre todos los dispositivos IDE de los que el núcleo está al tanto. Hay un subdirectorio por cada dispositivo (p.ej. disco duro) que contiene los siguientes ficheros:

cache	La caché.
capacity	Capacidad del medio.
driver	Controlador y versión.
geometry	Geometría física y lógica.
identify	Bloque de identidad del dispositivo.
media	Tipo de medio.
model	Identificador del dispositivo.
settings	Configuración del dispositivo.
smart_thresholds	IDE disk management thresholds
smart_values	DE disk management values

- Información de red en /proc/net.

Este directorio sigue la tendencia anterior. La tabla 1.3 lista los ficheros y su significado.

arp	Tabla de ARP del núcleo.
Dev	Dispositivos de red con sus estadísticas.
dev_mcast	Lista los grupos multicast Layer2 a los que un dispositivo está escuchando (índice de interfaz, etiqueta, número de referencias, número de direcciones enlazadas).
dev_stat	Estado del dispositivo de red.
ip_fwchains	Enlaces de cadenas del cortafuegos (firewall chain linkage).
ip_fwnames	Cadenas del cortafuegos.
ip_masq	Directorio que contiene las tablas de enmascaramiento.
ip_masquerade	Tabla de enmascaramiento mayor (major masquerading table).
netstat	Estadísticas de red.
raw	Estadísticas del dispositivo en bruto.
route	Tabla de enrutado del núcleo.
rpc	Directorio que contiene información de RPC.
rt_cache	Caché de enrutado.
snmp	Datos sobre SNMP.
sockstat	Estadísticas de sockets.
tcp	Sockets TCP.
tr_rif	Tabla de enrutado RIF para Token Ring.
udp	Sockets UDP.
unix	Sockets del dominio UNIX.
wireless	Datos del interfaz inalámbrico (Wavelan, etc.).
igmp	Direcciones multicast IP a las que este host se ha unido.
psched	Parámetros del planificador de paquetes globales. (Global packet scheduler parameters)
netlink	Lista de sockets PF_NETLINK.
ip_mr_vifs	Lista de interfaces virtuales multicast.
ip_mr_cache	Lista del caché de enrutado multicast.
udp6	Sockets UDP (IPv6).
tcp6	Sockets TCP (IPv6).
raw6	Estadísticas del dispositivo en bruto(raw) (IPv6).
igmp6	Direcciones multicast IP a las que se ha unido este host (IPv6).
if_inet6	Lista de direcciones de interfaces IPv6.
ipv6_route	Tabla de enrutado del núcleo para IPv6.

rt6_stats	Estadísticas globales de las tablas de enrutado de IPv6.
sockstat6	Estadísticas de sockets (IPv6).
snmp6	Datos sobre SNMP (IPv6).

Tabla 1.3: Información de red en /proc/net.

Puede usar esta información para ver qué dispositivos de red están disponibles en su sistema cuánto tráfico fue encaminado a esos dispositivos:

```
> cat /proc/net/dev
Inter-|Receive                                                |[...
  face |bytes        packets  errs  drop  fifo  frame  compressed
multicast|[...
    lo: 908188   5596     0    0    0     0          0         0 [...
  ppp0:15475140  20721   410    0    0   410          0         0 [...
  eth0:  614530   7085     0    0    0     0          0         1 [...

...] Transmit
...] bytes    packets errs drop fifo colls carrier compressed
...]  908188     5596    0    0    0     0       0          0
...] 1375103    17405    0    0    0     0       0          0
...] 1703981     5535    0    0    0     3       0          0
```

- Información SCSI.

Si tiene un adaptador de host SCSI, encontrará subdirectorios en /proc/scsi que llevan por nombre el controlador del adaptador. También puede ver una lista de todos los dispositivos SCSI reconocidos en /proc/scsi:

```
>cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: QUANTUM  Model: XP34550W        Rev: LXY4
  Type:   Direct-Access                   ANSI SCSI revision: 02
Host: scsi0 Channel: 00 Id: 01 Lun: 00
  Vendor: SEAGATE  Model: ST34501W        Rev: 0018
  Type:   Direct-Access                   ANSI SCSI revision: 02
Host: scsi0 Channel: 00 Id: 02 Lun: 00
  Vendor: SEAGATE  Model: ST34501W        Rev: 0017
  Type:   Direct-Access                   ANSI SCSI revision: 02
Host: scsi0 Channel: 00 Id: 04 Lun: 00
  Vendor: ARCHIVE  Model: Python 04106-XXX Rev: 703b
  Type:   Sequential-Access               ANSI SCSI revision: 02
```

El directorio bajo el nombre de cada controlador contiene un fichero por cada adaptador encontrado en el sistema. Estos ficheros contienen información sobre el dispositivo, como la IRQ y el rango de direcciones de E/S:

>cat /proc/scsi/ncr53c8xx/0
General información:
Chip NCR53C875, device id 0xf, revision id 0x4
IO port address 0xec00, IRQ número 11
Synchronous period factor 12, max commands per lun 4

- Información de puerto paralelo en /proc/parport.

El directorio /proc/parport contiene información sobre los puertos paralelos de su sistema. tiene un subdirectorio por cada puerto, cuyo nombre es el número del puerto (0,1,2,...).

Este directorio contiene cuatro ficheros:

autoprobe	Resultados del sondeo automático para este puerto.
devices	Módulos de dispositivo conectados.
hardware	Información del hardware (tipo de puerto, E/S del puerto, DMA, IRQ, etc.)
irq	Interrupción usada, si la hay.

- **Información de TTY en /proc/tty.**

Se puede encontrar información sobre las tty's disponibles y las que están siendo usadas, en /proc/tty. En este directorio encontrará entradas para los controladores y las disciplinas de la línea, como se muestra en la tabla inferior:

drivers	Lista de controladores y su uso.
ldiscs	Disciplinas de línea registradas.
driver/serial	Estadística de uso y estado de las líneas de una sola tty.

Para ver qué tty's están actualmente en uso, puede mirar sencillamente dentro del fichero /proc/tty/drivers:

>cat /proc/tty/drivers
pty_slave /dev/pts 136 0-255 pty:slave
pty_master /dev/ptm 128 0-255 pty:master
pty_slave /dev/ttyp 3 0-255 pty:slave
pty_master /dev/pty 2 0-255 pty:master
serial /dev/cua 5 64-67 serial:callout
serial /dev/ttyS 4 64-67 serial
/dev/tty0 /dev/tty0 4 0 system:vtmaster
/dev/ptmx /dev/ptmx 5 2 system
/dev/console /dev/console 5 1 system:console
/dev/tty /dev/tty 5 0 system:/dev/tty
unknown /dev/tty 4 1-63 console

- Leyendo y modificando parámetros del núcleo.

Una parte muy interesante /proc es el directorio /proc/sys. No sólo proporciona información, sino que además le permite cambiar parámetros de dentro del núcleo. Sea muy cuidadoso cuando intente esto. Puede optimizar su sistema, pero también puede dejarlo "fuera de combate". Nunca juegue a tocar los parámetros del núcleo de un sistema en producción. Instale una máquina de desarrollo y haga las pruebas en ella para asegurarse de que todo funciona de la manera que desea. Puede no tener más alternativa que reiniciar la máquina una vez que se ha cometido un error.

Para cambiar un valor, simplemente escriba el nuevo valor en el fichero. Más abajo se da un ejemplo en la sección de datos del sistema de ficheros. Necesita ser root para hacer esto. Puede crear su propio script de inicio para conseguir que todo esto se haga cada vez que su sistema arranque.

Los ficheros en /proc/sys pueden usarse para ajustar y controlar distintos aspectos generales de la operación del núcleo Linux. Ya que algunos ficheros pueden interferir de forma inadvertida con su sistema, es aconsejable leer tanto la documentación como las fuentes antes de hacer realmente ajustes. En cualquier caso, sea muy cuidadoso cuando escriba en cualquiera de estos ficheros. Las entradas en /proc pueden cambiar ligeramente entre los núcleos 2.1.* y el núcleo 2.2, así que revise la documentación si le surge alguna duda. Encontrará la documentación en el directorio /usr/src/linux/Documentation/sys. Este capítulo está en gran parte basado en la documentación incluída en los núcleos pre-2.2. Gracias a Rick van Riel por proporcionar toda esa información.

- /proc/fs - Datos del sistema de ficheros.

Este subdirectorio contiene información específica del sistema de ficheros, manipuladores de ficheros, inodos, entradas de directorio (llamadas "dentry") y cuotas. Actualmente, estos ficheros están en /proc/sys/fs:

dentry-state

Estado del caché de directorios. Ya que las entradas de directorio son alojadas y desalojadas dinámicamente, este fichero da información sobre el estado actual. Mantiene seis valores, de los que los dos últimos son siempre cero y no se usan. Los otros cuatro significan:

nr_dentry	Parece ser cero todo el tiempo.
nr_unused	Número de entradas del caché sin usar.
age_limit	Tiempo en segundos tras el que una entrada puede reclamarse, cuando la memoria escasee.
want_pages	Interno.

dquot-nr y dquot-max

El fichero dquot-max muestra el máximo número de entradas de cuota de disco en el caché.

El fichero dquot-nr muestra el número de entradas de cuotas de disco alojadas y el número de entradas de cuota de disco libres.

Si el número de cuotas de disco libres es muy baja y tiene un gran número de usuarios simultáneos en el sistema, podría querer aumentar el límite.

file-nr y file-max

El núcleo aloja manipuladores de fichero de forma dinámica, pero al menos por el momento no los libera de nuevo.

El valor en file-max denota el número máximo de manipuladores de fichero de forma dinámica que el núcleo de Linux alojará. Si obtuviese un montón de mensajes de error referentes a escasez de manipuladores de fichero (running out of file handles), podría querer aumentar este límite. El valor por defecto es 4096. Para cambiarlo, simplemente escriba el nuevo valor en el fichero:

```
# cat /proc/sys/fs/file-max
4096
# echo 8192 > /proc/sys/fs/file-max
# cat /proc/sys/fs/file-max
8192
```

Este método de modificación es prácticamente el mismo para todos los parámetros del núcleo susceptibles de ser modificados; simplemente introduzca el nuevo valor en el fichero correspondiente.

Los tres valores de file-nr denotan el número de manipuladores de fichero alojados, los usados, y el número máximo de ellos. Cuando el número de manipuladores de fichero alojados se acerque al máximo, pero el número de los que estén siendo usados sea mucho menor, habrá encontrado un pico en el uso de manipuladores y no necesitará incrementar el máximo.

Sin embargo, hay además un límite en el número de ficheros abiertos por proceso que, por desgracia, no se puede cambiar tan fácilmente. Su valor por defecto es 1024. Para cambiarlo tiene que editar los ficheros limits.h y fs.h del directorio /usr/src/linux/include/linux. Cambie la definición de NR_OPEN y recompile el núcleo.

inode-state, inode-nr y inode-max

Como sucede con los manipuladores de fichero, el núcleo aloja las estructuras de inodo dinámicamente, pero no las puede liberar de inmediato.

El valor en inode-max denota el máximo número of manipuladores de inodo. Este valor debería ser 3 ó 4 veces más grande que el valor de file-max, ya que también se necesitan estructuras de inodo para manejar stdin, stdout, y los sockets de red. Si se queda sin inodos de una forma regular, debería incrementar este valor.

DISPOSITIVOS

/dev

Echémosle un vistazo de nuevo con ls -F. Los "ficheros" en /dev son conocidos como controladores de dispositivo (device drivers)_son usados para acceder a los dispositivos del sistema y recursos, como discos duros, modems, memoria, etc. Por ejemplo, de la misma forma que puede leer datos de un fichero, puede leerla desde la entrada del raton leyendo /dev/mouse. Los ficheros que comienzan su nombre con fd son controladores de disqueteras. fd0 es la primera disquetera, fd1 la segunda. Ahora, alguien astuto se dará cuenta de que hay mas controladores de dispositivo para disqueteras de los que hemos mencionado. Estos representan tipos específicos de discos. Por ejemplo, fd1H1440 accederá a discos de 3.5" de alta densidad en la disquetera 1. Aqui tenemos una lista de algunos de los controladores de dispositivo mas usados. Nótese que incluso aunque puede que no tenga alguno de los dispositivos listados, tendrá entradas en dev de cualquier forma.

- /dev/console hace referencia a la consola del sistema_ es decir, al monitor conectado directamente a su sistema.

- Los dispositivos /dev/ttyS y /dev/cua son usados para acceder a los puertos serie. Por ejemplo, /dev/ttyS0 hace referencia a "COM1" bajo MS-DOS. Los dispositivos /dev/cua son "callout", los cuales son usados en conjunción con un módem.

- Los nombres de dispositivo que comienzan por hd acceden a discos duros. /dev/hda hace referencia a la totalidad del primer disco duro, mientras que /dev/hda1 hace -referencia a la primera partición en /dev/hda.

- Los nombres de dispositivo que comienzan con sd son dispositivos SCSI. Si tiene un disco duro SCSI, en lugar de acceder a el mediante /dev/hda, deberá acceder a /dev/sda. Las cintas SCSI son accedidas vía dispositivos st y los CD-ROM SCSI vía sr.

- Los nombres que comienzan por lp acceden a los puertos paralelo. /dev/lp0 hace referencia a "LPT1" en el mundo MS-DOS.

- /dev/null es usado como "agujero negro" cualquier dato enviado a este dispositivo desaparece. >Para que puede ser útil esto?. Bien, si desea suprimir la salida por pantalla de una orden, podría enviar la salida a /dev/null. Hablaremos mas sobre esto después.

- Los nombres que comienzan por /dev/tty hacen referencia a "consolas virtuales" de su sistema (accesibles mediante las teclas |_alt-F1_,||_alt-F2_,|etc). /dev/tty1 hace referencia a la primera VC, /dev/tty2 a la segunda, etc.

- Los nombres de dispositivo que comienzan con /dev/pty son "pseudo-terminales". Estos son usados para proporcionar un "terminal" a sesiones remotas. Por ejemplo, si su maquina esta en una red, telnet de entrada usara uno de los dispositivos /dev/pty.

Dispositivos y particiones en Linux

Muchas distribuciones necesitan que se creen a mano las particiones de Linux utilizando el programa fdisk. Otras pueden crearlas automáticamente. En cualquier caso, usted debe conocer lo siguiente acerca de los nombres para los dispositivos y las particiones en Linux.

Bajo Linux, los dispositivos y las particiones tienen nombres muy distintos a los utilizados en otros sistemas operativos. Bajo MS-DOS, las disqueteras se identifican como A: y B:, mientras que las particiones del disco duro se identifican como C:, D, etc. Bajo Linux, la denominación es algo diferente.

Los manejadores de dispositivos, que se encuentran en el directorio /dev, se usan para comunicar con los dispositivos de su sistema (como discos duros o ratones). Por ejemplo, si usted tiene un raton en su sistema, lo puede acceder a traves del manejador /dev/mouse. Las disqueteras, discos duros y particiones tienen cada uno un manejador propio. No se preocupe acerca de la interfaz del manejador por ahora; solo es importante entender como son nombrados los dispositivos con el fin de poderlos usar.

La tabla 2.1 lista los nombres de diversos manejadores.

Dispositivo	Nombre
Primera disquetera (A:)	/dev/fd0
Segunda disquetera (B:)	/dev/fd1
Primer disco duro (todo el disco)	/dev/hda
Primer disco duro, partición primaria 1	/dev/hda1
Primer disco duro, partición primaria 2	/dev/hda2
Primer disco duro, partición primaria 3	/dev/hda3
Primer disco duro, partición primaria 4	/dev/hda4
Primer disco duro, partición lógica 1	/dev/hda5
Primer disco duro, partición lógica 2	/dev/hda6
..	
.	
Segundo disco duro (todo el disco)	/dev/hdb
Segundo disco duro, partición primaria 1	/dev/hdb1
..	
.	
Primer disco duro SCSI (todo el disco)	/dev/sda
Primer disco duro SCSI, partición primaria 1	/dev/sda1
..	
.	
Segundo disco duro SCSI (todo el disco)	/dev/sdb
Segundo disco duro SCSI, partición primaria 1	/dev/sdb1
..	
.	

Tabla 2.1: Nombres de discos y particiones en Linux

Algunas notas acerca de esta tabla. Observe que /dev/fd0 corresponde a la primera disquetera
(A: bajo MS-DOS) y que /dev/fd1 corresponde a la segunda (B:).

Además, los discos duros SCSI se nombran de manera diferente a otros discos. Los IDE, MFM y RLL se acceden a traves de los dispositivos /dev/hda, /dev/hdb, etc. Las particiones de /dev/hda son /dev/hda1, /dev/hda2, etc. Sin embargo, los dispositivos SCSI se nombran con /dev/sda, /dev/sdb, etc., y las particiones con /dev/sda1, /dev/sda2, etc.

Aqui tenemos un ejemplo. Supongamos que usted tiene un disco duro IDE con 3 particiones primarias. Las dos primeras son para MS-DOS, y la tercera es extendida y contiene dos particiones lógicas, ambas para ser usadas con Linux. Los dispositivos quedarían representados con:

Primera partición MS-DOS (C:)	/dev/hda1
Segunda partición MS-DOS (D:)	/dev/hda2
Partición extendida	/dev/hda3
Primera partición lógica de Linux	/dev/hda5
Segunda partición lógica de Linux	/dev/hda6

Observe que nos hemos saltado /dev/hda4, ya que corresponde a la cuarta partición primaria, que no existe en el ejemplo. Las particiones lógicas se nombran de forma consecutiva partiendo de /dev/hda5.

DETERMINAR EL HARDWARE SOPORTADO

Esto es quizá lo más importante de todo. Todas las grandes virtudes y ventajas de Linux no servirán de mucho si la PC en donde se instalará no posee el hardware adecuado y en buen estado. Si el hardware y/o periféricos tienen problemas en MS-DOS/Windows, o alguna de las funciones no trabaja adecuadamente, será muy difícil que lo haga en Linux. Es entonces conveniente que revisemos varios puntos, además de la documentación que acompaña a cada versión.

Es indispensable determinar al menos el modelo exacto de la tarjeta de video y, si existiese, el modelo de la tarjeta SCSI.

El microprocesador.

Linux trabajará con cualquier microprocesador 80386, 80486, 80586 (Pentium y equivalentes) y 80686 (Pentium II y equivalentes), etc..., siempre que estos estén en buen estado y que sean originales. La siguiente es una lista de los microprocesadores que le sugerimos para poder utilizar Linux con total funcionalidad:

- AMD
- AMD K5
- AMD K6 (algunos modelos anteriores tienen errores. Puede ser conveniente deshabilitar el cache interno).
- AMD K6-II
- -AMD K6-III
- AMD Athlon
- AMD K7
- Intel
- Pentium
- Pentium Pro
- Pentium MMX
- Pentium II
- Pentium III
- Pentium IV
- Merced
- Microprocesadores Cyrix 80686 y MII
- Microprocesadores Winchip

Asegúrese de que el microprocesador de la PC sea original y con garantía. Este es un punto poco comentado, pero es muy trascendental. Si usted posee una PC de marca (Compaq, Dell, Hewlett Packard, IBM, etc.) no tiene de que preocuparse. Si usted, por el contrario, posee una PC ensamblada (sin marca), asegúrese de que se le proporcionó el certificado de autenticidad del Microprocesador con la documentación que los ensambladores están obligados a proporcionarle. Si no se le proporcionó dicho certificado, usted mismo puede averiguar si el Microprocesador de su PC es "bueno":

- Descargue la electricidad estática de su cuerpo tocando un objeto metálico grande.
- Retire los tornillos en la parte posterior de la PC y retire la tapa.
- Localice el microprocesador, normalmente este posee un ventilador pequeño.
- Revise si el ventilador posee un holograma o logotipo del fabricante (Intel, AMD, etc.).
- Si se trata de un microprocesador OEM, este no incluye ventilador de fábrica, levante la pequeña palanca localizada a un costado del soquet y retire el microprocesaro. Este deberá tener una etiqueta adherible en la parte inferiror con el número de serie impreso.
- Coloque todo en su lugar y cierre el gabinete.

Si el ventilador del microprocesador no tiene un holograma o logotipo del fabricante, o bien usted ha observado un ventilador color azul, lo más probable es que este microprocesador sea del tipo "a granel".

¿Que es un microprocesador "a granel"? Un "desecho de fabrica" o bien un microprocesador reconstruido. No debe confundirse con un microprocesador OEM, que se vende sin ventilador, sin manual y sin caja, pero tiene garantía por parte del fabricante.

Un microprocesador "a granel" jamás trabajará con el mismo rendimiento de un microprocesador "en caja" u OEM, ambos con garantía. Si no lo cree, compre un microprocesador "a granel" e intente obtener servicio de soporte por parte del fabricante...

Tarjeta de video.

Cada versión de Linux proporciona listas de hardware soportado. Asegúrese de verificar que el hardware de la PC este en las mencionadas listas. Si usted tiene el CD de instalación, consulte el archivo HOWTO referente al Hardware. La tarjeta de video es quizá uno de los más importantes, si usted desea utilizar el modo gráfico X Window.

Al momento de configurar la tarjeta de video, Xconfigurator le preguntará cual modelo utiliza. Debe especificar el modelo exacto, no trate de configurar la tarjeta de video como un modelo similar o parecido o estará perdiendo el tiempo. Si acaso la tarjeta de video no estuviese especificada, podrá especificar una configuración que le permitirá trabajar en modo gráfico, siempre que conozca las características de la tarjeta, como memoria (VideoRAM) y tipo, resolución del monitor y rangos de refresco vertical y horizontal de este último.

Tarjeta de audio.

La tarjeta de audio, quizá algo de entre lo más difícil de configurar, es importante, a menos que no le interese contar con soporte de audio. Cabe aclarar que las tarjetas de audio o video baratas y de calidad regular, como la tarjeta de audio CMI8330, suelen no ser totalmente funcionales o suelen presentar defectos de manufactura que se evidencian claramente en Linux.

Algunos modelos de tarjetas de audio, como la misma CMI8330, pueden requerir se editen manualmente los archivos "config.modules y "isapnp.conf" del directorio "/etc", y tal vez sea necesario compilar el Kernel de una forma específica. a continuación presentamos una lista de algunas tarjetas de audio soportadas por Linux:

- 6850 UART MIDI
- Adlib (OPL2)
- Audio Excell DSP16
- Aztech Sound Galaxy NX Pro
- CMI8330 (C-Media) (puede necesitar compilar el kernel de forma específica y modificar manualmente "Isapnp.conf" y "conf.modules")
- Crystal CS4232/CS4236 (PnP)
- Ensoniq SoundScape
- Gravis Ultrasound
- Gravis Ultrasound 16-bit
- Gravis Ultrasound MAX
- Gravis Ultrasound ACE (El puerto midi y grabación de audio no funcionarán)
- Gravis Ultrasound PnP (con RAM)
- Logitech SoundMan Games (SBPro, soporte 44kHz estero)
- Logitech SoundMan Wave (Jazz16/OPL4)
- Logitech SoundMan 16 (PAS-16 compatible)
- MediaTriX AudioTriX Pro
- Media Vision Premium 3D (Jazz16)
- Media Vision Pro Sonic 16 (Jazz)
- Media Vision Pro Audio Spectrum 16
- Media Vision Pro Audio Studio 16
- Microsoft Sound System (AD1848)
- OAK OTI-601D (Mozart)
- OPTi 82C924/82C925. (Necesitan el controlador MSS e "isapnptools")
- OPTi 82C928/82C929 (MAD16/MAD16 Pro/ISP16/Mozart)
- OPTi 82C931
- Sound Blaster
- Sound Blaster Pro
- Sound Blaster 16
- Sound Blaster 32/64/AWE (Debe configurarse como si fuese Sound Blaster 16)
- Sound Blaster AWE63/Gold y tarjetas 16/32/AWE PnP (nota: se necesita utilizar "isapnptools" para activarlas)
- Turtle Beach Wavefront (Maui, Tropez)
- Wave Blaster
- Tarjetas basadas sobre AudioDrive chips ESS (688, 1688)
- AWE32/64
- MPU-401 MIDI

Tarjetas de Red.

La mayoría de las tarjetas de red son soportadas por Linux, salvo algunas excepciones las cuales se mencionan en el archivo HOWTO acerca del hardware, en el CD de instalación.

Modems.

Estos trabajarán bien con Linux, mientras no sean del tipo "exclusivamente con Windows", como son los Winmodems y softmodems, es decir modems de muy bajo costo, y rendimiento apenas aceptable, que trabajan con principalmente con software (y no hardware) y bajo el ambiente de Windows. Algunos ejemplos de softmodems o Winmodems y modems que no trabajan con Linux son los Pctel y Motorola internos, así que será mejor evitarlos y ahorrarse disgustos.

De todas formas poco a poco las empresas y los colaboradores van creando drives. Los podemos encontrar en:

http://www.linmodems.org

En general cualquier modem que se conecte utilizando un puerto de comunicación real (y no virtual, como con los softmodems y Winmodems) trabajará con Linux. Los modems que utilizan slot ISA, son fáciles de configurar. Los modems que utilizan un slot PCI suelen tener problemas, debido a que la mayoría son softmodems o Winmodems, solo será posible configurar los modems reales. Irónicamente, los modems no-PnP (Plug & Play) y los modems externos son los que trabajan mejor y los que resultan más fáciles de configurar. Si le interesa un modem interno que trabaje de forma excelente, le sugiero los Hayes e IBM no-PnP internos.

El modem podrá configurarlo cuando haya concluido la instalación utilizando el comando "minicom" o "isapnptools" modo de texto o utilizando "Linuxconf" en modo X Window.

Monitores.

Casi todos los monitores pueden trabajar con Linux. En caso de que su monitor no se encuentre especificado en la base de datos de Xconfigurator, deberá proporcionar los rangos de refresco vertical u horizontal y resolución máxima de su monitor. Consulte el manual del monitor para obtener dichos datos, de lo contrario, si especifica valores incorrectos (valores de refresco más altos de los especificados para el monitor), **puede dañar el monitor irremediablemente**. Recuerde que, de ser necesario, puede especificar rangos menores sin riesgo alguno. Ejemplo: si su monitor posee un rango de refresco vertical 50-120, puede especificar un rango menor como 50-100, pero **nunca** 50-150.

Si lo desea, puede modificar desde X Window el archivo "XF86Config" del directorio "/etc/X11" y especificar los rangos exactos de su monitor. Es conveniente cerrar la sesión X Window y reiniciar la misma para que surtan efecto los cambios. Nuevamente, no especifique un rango mayor al que realmente posee e monitor o puede dañar irremediablemente este.

Unidades de disco duro.

Casi todos los tipos de disco duro (IDE y SCSI) trabajarán bien con Linux.

Unidades de disco extraibles y similares.

Las últimas versiones del Kernel de Linux proporcionan soporte instantáneo (o casi instantáneo) para unidades Jaz y Zip drive y otras más de su tipo, tanto para las versiones SCSI como para las ATAPI, IDE y puerto paralelo. Incluso, si le interesa, puede encontrar software hecho para trabajar de la misma forma en que se hace en Windows (control de protección lectura/escritura, formato, iconos, etc.).
En estos URLs puede encontrar software gratuito para Jaz y Zip Drive:

http://www.torque.net/~campbell/
http://www.scripps.edu/~jsmith/jazip/

Para configurar cada unidad de disco o cinta extraible, y cualquier otro sistema de archivos que posea o desee incluir, puede utilizar "Linuxconf" en el modo X Windows o bien editar el contenido de el archivo "fstab" del directorio "/etc". Ejemplo:

```
/dev/hda7       /       ext2        defaults        1       1
/dev/hda8       /home       ext2        defaults        1       2
/dev/hda9       swap        swap        defaults        0       0
#
#                                                               Floppy
/dev/fd0  /mnt/floppy  vfat  user,exec,dev,suid,rw,noauto  0  0
#
#       Cd-rom      /dev/hdc        /mnt/cdrom      iso9660
user,exec,dev,suid,ro,noauto                0               0
#
none        /proc       proc        defaults        0       0
none       /dev/pts       devpts       mode=0622       0       0
#
#
# Discos duros del sistema que utiliza con MS-DOS/Windows
95/98
/dev/hda1   /mnt/disco-c   vfat   user,exec,dev,suid,rw   1   1
```

```
/dev/hda5   /mnt/disco-d   vfat   user,exec,dev,suid,rw   1   1
/dev/hda6   /mnt/disco-e   vfat   user,exec,dev,suid,rw   1   1
#
#
#           Zip           drive          atapi          interno
#
/dev/hdb4 /mnt/zip vfat user,exec,dev,suid,rw,noauto 1
1
```

No olvide especificar estos valores, o puede llevarse algunas sorpresas:

user: significa que todos los usuarios pueden montar y desmontar esta unidad

rw: significa que la unidad es de lectura y escritura

noauto: significa que este sistema de archivos no se montará en le arranque. Esto es importante, ya que si no lo especifica, al momento de reiniciar, el sistema le indicará que no hay tal unidad disponible, a menos que haya un disco o cinta insertado.

xec: significa que pueden ejecutarse programas y comandos en esta unidad.

Escáners e impresoras.

Estos trabajarán bien con Linux siempre que, a igual que con los modems, no sean del tipo "exclusivamente Windows", como es la impresora Epson Stylus 400 (la mayoría del resto de los modelos de Epson trabajan bien con Linux) y otras más. Sin embargo, esto no significa que no pueda hacer uso de la impresora, ya que puede configurarla como "Genérica" de ser necesario. Lo mismo se aplica para los escáners.

USB

Para los elementos que se conectan al interfaz US, la comunidad Linux ha creado hotplug. Esta aplicación detecta y configura casi todos los componentes USB que existen. Toda la información la podemos encontrar en la página:

http://www.linux-usb.org

ARRANQUE DEL SISTEMA OPERATIVO

Otra tarea que de la que se hace cargo el administrador es arrancar y detener el sistema, controlando los servicios que se inician. ESware Linux utiliza el proceso de arranque init de Sys V; init es el primer proceso que se ejecuta en el sistema, es el más importante, del que dependen el resto de procesos del sistema.

INICIO DEL SISTEMA

EL PROCESO INIT

El núcleo ejecuta **init** al arrancar; Este programa, ahora como proceso, cargará los subprocesos necesarios para la puesta en marcha del sistema. Cuando **init** ha terminado de cargarse vacía el subdirectorio **/tmp** y lanza **getty** que se encarga de permitir hacer ***login*** en el sistema a los usuarios.

Los niveles de ejecución *(**runleve**l*) determinan los servicios que tendremos disponibles en cada uno de ellos. Es una forma de tener diferentes modos de trabajo, cada uno de ellos con características bien definidas, en función del tipo de tarea a que estén orientados.

Existen siete niveles de ejecución, que están numerados del cero al seis, más otro denominado con la letra «S» (tiene un alias con la letra «s», que es básicamente igual a el nº 1).

Los niveles de ejecución que manejaremos y una descripción de para qué están definidos se puede ver en la siguiente tabla:

Nivel de ejecución	***Modo***
0	Detener el sistema
1	Monousuario, sin soporte de red
2	Multiusuario, sin soporte de red
3	Modo multiusuario completo
4	Sin uso. Recomendado para pruebas
5	multiusuario completo en entorno gráfico
6	Reiniciar el sistema

Init necesita un fichero de configuración para saber exactamente lo que tiene que hacer. Este fichero es **/etc/inittab** y contiene información sobre el nivel a ejecutar por defecto, previsión sobre lo que hacer ante determinadas situaciones, describe qué procesos se inician en la carga y durante la operación normal.

Las entradas del fichero ***/etc/inittab*** tienen el siguiente formato:

id:niveles_ejecución:acción:proceso

- **id**: Una secuencia única de 1 a 4 caracteres que identifican la entrada de ***inittab***

- **niveles_ejecución**: Lista de niveles de ejecución para los que se llevarán a cabo las acciones definidas a continuación en la línea.

- **acción**: La acción se llevará a cabo.

- **proceso**: El proceso a ejecutar.

Para que una línea sirva para varios niveles de ejecución, el campo ***niveles_ejecución*** tiene que incluirlos. Por ejemplo, 135 indica que el proceso se iniciará en los niveles 1, 3 y 5. Cuando se cambia de un nivel de ejecución a otro, los procesos en ejecución que no estén definidos en el nuevo nivel se matan.

Las acciones que podemos definir, más habitualmente, en el campo acción son:

initdefault: Especifica el nivel de ejecución por defecto al arrancar el sistema. El campo proceso se ignora.

Respawn: El proceso se reiniciará cuando termine.

once: El proceso se ejecutará una sola vez cuando se entre en el nivel de ejecución especificado.

wait: El proceso se iniciará una vez cuando se entre en el nivel de ejecución e init esperará a su terminación.

boot: El proceso se ejecutará durante el arranque del sistema. El campo *niveles_ejecución* se ignora.

bootwait: El proceso se ejecutará durante el arranque del sistema, mientras init espera su terminación. El campo niveles_ejecución se ignora.

sysinit: El proceso se ejecutará durante el arranque del sistema, antes que cualquier entrada boot o bootwait. El campo *niveles_ejecución* se ignora.

powerwait: El proceso se ejecutará si init recibe una señal SIGPWR, que indica algún problema con la alimentación eléctrica. Init esperará que el proceso termine.

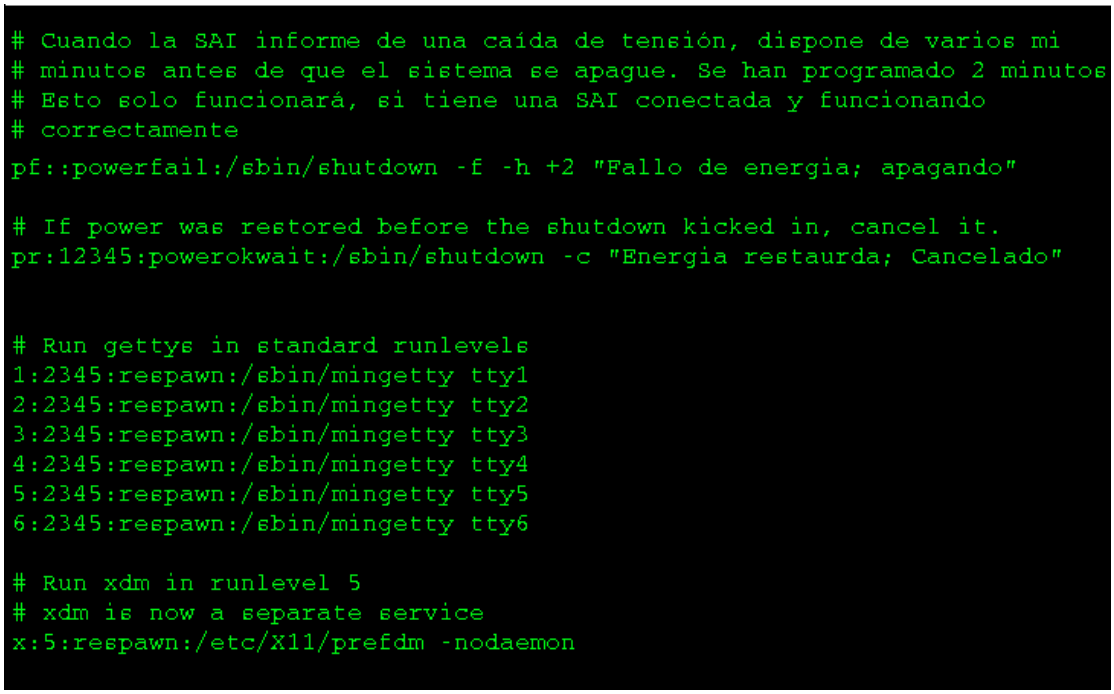
powerfail: Como ***powerwait***, excepto que init no espera a que termine el proceso.

powerokwait: El proceso se ejecutará si init recibe la señal SIGPWR, con la condición de que haya un fichero llamado ***/etc/powerstatus*** que contenga la palabra OK. Esto significa que se ha restablecido la alimentación eléctrica.

ctrlaltdel: Especifica qué proceso se ejecutará al pulsar la combinación de teclas [Ctrl+Alt+Supr]. Normalmente, reiniciar la máquina.

```
#
# inittab       Este fichero describe como arrancará el sistema el proceso INIT
#               en cada nivel de ejecución.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modificado para RHS Linux por Marc Ewing and Donnie Barnes
#               y por Sergio Rua <srua@esware.com>
#

# Nivel de ejecución por defecto. Los niveles que usa Esware Linux son:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

/etc/inittab



```
# Cuando la SAI informe de una caída de tensión, dispone de varios mi
# minutos antes de que el sistema se apague. Se han programado 2 minutos
# Esto solo funcionará, si tiene una SAI conectada y funcionando
# correctamente
pf::powerfail:/sbin/shutdown -f -h +2 "Fallo de energia; apagando"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Energia restaurda; Cancelado"


# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

/etc/inittab

Uno de los *scripts* más importantes en el arranque del sistema es ***/etc/rc.d/rc.sysinit***. Es el primer script que init encuentra y ejecuta. En él están definidas funciones como:

- Inicio y activación del espacio de intercambio. *(swap*)

- Configuración de la red.

- Especificación de variables del sistema.

- Comprobación y montaje de los sistemas de archivos.

- Inicialización de puertos serie.

- Carga los módulos del *kerne*l.

- Establecimiento de cuotas de usuarios.

- Ajuste del reloj del sistema.

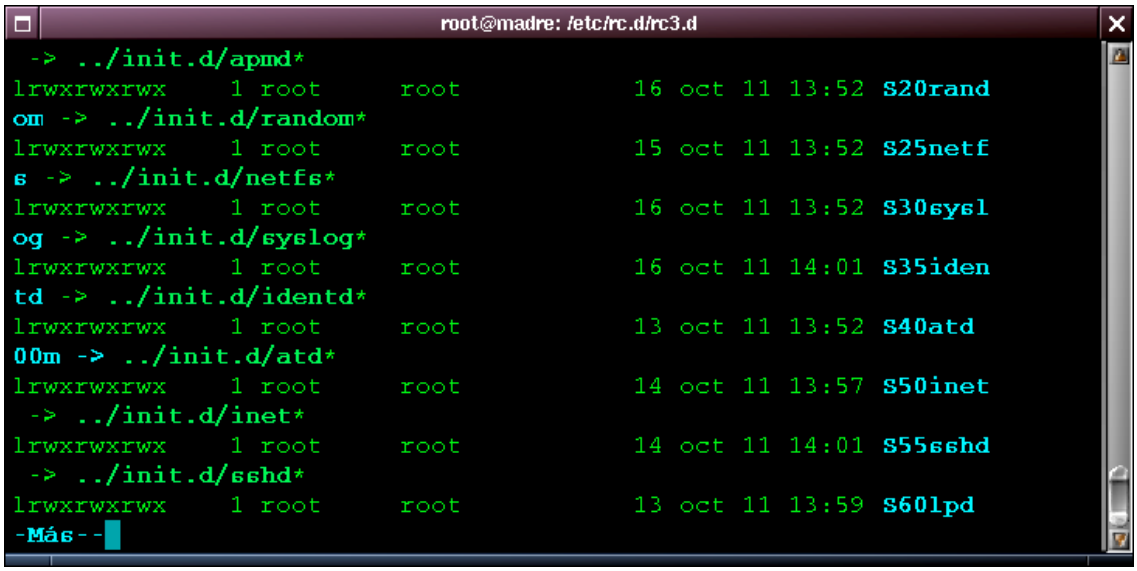
El último *script* en ejecutarse es ***/etc/rc.d/rc.local***. En este fichero podremos poner inicializaciones especificas del sistema, aunque su propósito inicial es controlar los servicios de red.

DIRECTORIOS DIRECTAMENTE IMPLICADOS

El directorio ***rc.d*** es de vital importancia para el arranque del sistema. Tiene una estructura bien definida ya que existe un directorio para cada nivel de ejecución, identificado con el mismo número que el nivel, y en cada uno de estos directorios se indican qué servicios se iniciarán o detendrán al entrar en ese nivel de ejecución.

El directorio ***init.d*** contiene los *scripts* que lanzarán o detendrán los servicios que tengamos disponibles en nuestro equipo. La normalmente es suficiente invocar al servicio por su nombre y pasarle uno de los siguientes argumentos: ***start***, ***stop***, ***status***, ***restart*** o reload .

Los directorios numerados para cada *runlevel* contienen enlaces simbólicos que apuntan a los scripts del directorio ***init.d***. Veamos un muestra, esto es parte del directorio ***rc3.d***:



rc3.d

Observemos con atención como se llaman los enlaces, cada uno de ellos tiene el nombre del *script* al que están asociados. Los que empiezan con una «S» («S» de *star*t) define si el servicio se inicia, y otros con una «K»define si el servicio se detiene («K» de *ki*ll). El número que lucen es simplemente una facilidad para ordenar y que no tiene mayor relevancia.

Lo realmente ventajoso de este sistema es que, en primer lugar, no se repiten los *scripts* en cada directorio de runlevel, si no que permanecen en un único lugar bien definido, el directorio init.d, y en segundo lugar, la modificación a realizar si lanzamos un servicio o no, en un runlevel determinado, es tan sencilla como cambiar el nombre del enlace a el servicio en cuestión.

Si queremos que se inicie bastará con asegurarse de que su nombre empieza por una «S» y en caso contrario, ósea que en ese nivel de ejecución no se ofrezca el servicio, pondremos el nombre empezando por una «K». Así de sencillo.

Vuelva a mirar el fichero ***/etc/inittab*** de más arriba. Hay una línea para cada nivel de ejecución que tiene como proceso «rc» con los números de los runlevels como parámetro. El ***script* rc** se encarga de reiniciar el sistema en un nivel de ejecución distinto.

Otra ventaja es el control que tenemos sobre los servicios del sistema. Independiente del estado en el que estén, los podemos lanzar, detener, reiniciar, etc. sobre la marcha, sin necesidad de reiniciar la máquina, ni nada parecido.

EL COMANDO INIT

Podemos ejecutar init desde línea de comandos con alguno de los siguientes argumentos:

0, 1, 2, 3, 4, 5, 6: Para cambiar al nivel de ejecución especificado.

Q, q: Si queremos que init relea el fichero **/etc/inittab**.

S, s: Entra en modo monousuario .

U, u: Reejecuta init respetando el estado actual. No se relee el fichero ***/etc/inittab***.

CREACIÓN DE DISCOS DE ARRANQUE DEL SISTEMA

Para crear discos de arranque del sistema, necesitamos tener las imágenes de los discos para poderlos volcar a los diskettes.

Una imagen de disco es una copia binaria exacta del soporte físico del cual se ha hecho la copia.

MkBootDisk y MkBoot

Crea un diskette de arranque apropiado al sistema que tenemos instalado.

[madre@madre~]$ mkbootdisk -device /dev/fd0 <kernel>

donde dice <kernel> se escribe lo que nos devuelve el comando uname ***-r***

El comando ***mkboot*** también crea un diskette de rescate apropiado. No necesita argumentos, pero a veces es necesario que en el directorio raíz (**/**) exista un enlace simbólico que apunte a la versión del kernel activo que está en el directorio /boot. La sintaxis es:

mkboot

Nos responderá algo parecido a esto: Insert a floppy diskette into your boot drive, and press <Return>. Debemos entonces introducir un diskette en la unidad y pulsar intro.

Compilar el Núcleo

El núcleo (o kernel, se usarán los dos términos indistintamente) es en realidad lo más importante del sistema; Es el encargado de interactuar con el hardware de nuestro equipo.

Cuando un usuario ejecuta programas, es el kernel el responsable de manejar la información para que el hardware se ponga a trabajar, ya sea comunicarse con la tarjeta de red o modem, gestionar el tiempo de uso de la CPU por cada programa (multitarea), la administración de memoria, etc.

Todas estas funciones definen el funcionamiento de la máquina y es muy interesante conocer la posibilidades que nos permiten «afinar» la manera en que se va a comportar.

Versiones

El núcleo está en continuo desarrollo y se actualiza con bastante frecuencia, por lo que es imprescindible saber que versión tenemos en nuestro equipo, que hardware soporta, y si realmente necesitaremos actualizarlo o simplemente re-compilar el que ya tenemos para optimizar sus prestaciones.

El comando ***uname -a*** muestra información de nuestro equipo, entre la cual

se encuentra el número de la versión que estamos utilizando.

Los números de versión del núcleo se representa siguiendo un patrón que aporta bastante información, es más o menos así: 2.4.16.

El primer número indica la versión, una nueva versión solo se realiza cuando el kernel ha experimentado grandes cambios.

El segundo número indica la actualización, a mayor valor, más reciente es el núcleo, ademas si este número es par, indica que esa versión es estable, (y estable en el núcleo quiere decir exactamente eso, estable.).

Sin embargo un número impar indica que es una versión de desarrollo y puede ser inestable. No quiere decir que vaya a ir mal necesariamente, pero se recomienda únicamente si se quiere experimentar o sí se necesita soporte urgente para algún tipo de dispositivo que se incluya en esta versión.

El último número es el de la revisión, indica el nivel de actualización que ha alcanzado esa versión.

Desde la llegada de la versión 2.x.x se trabaja con dos partes del núcleo diferentes, en primer lugar tenemos el núcleo en sí, (lo podemos llamar parte monolítica del núcleo) y en segundo lugar están los módulos.

La parte monolítica del núcleo es la que se carga en cada arranque y está activa en todo momento, sin embargo los módulos únicamente se activan cuando se requiere utilizar ese dispositivo en concreto.

Esto significa que podemos tener una serie de dispositivos soportados, pero que no están cargados en memoria durante todo el tiempo, si no que se incorporan cuando se hace una llamada a ese hardware.

Tenga en cuenta el uso de módulos en todas las opciones que pueden ayudarle a optimizar el núcleo, y por consiguiente el rendimiento del sistema. Pero preste también atención a determinadas opciones que obligatoriamente deberán ir en la parte monolítica, como puede ser el soporte del sistema de archivos ***ext2fs***, que se requiere desde al momento de arrancar.

EL CÓDIGO FUENTE

Para compilar el núcleo necesariamente debe tener en el sistema el código fuente. En el caso de no haberlo instalado, hágalo desde el CD-ROM del sistema o descárguelo del sitio oficial del núcleo:

http://www.kernel.org

Normalmente las fuentes se instalan en el directorio ***/usr/scr/linux***.

Si va a actualizar y quiere mantener el código anterior, mantenga cada versión en directorios diferentes (por ejemplo: **/usr/src/2.4.16** para la versión anterior y ***/usr/src/2.4.18*** para la nueva) y simplemente haga un enlace simbólico del directorio que tenga el código que desee compilar al directorio ***/usr/src/linux***.

Una vez instalado podemos ver un conjunto de directorios con todo el código necesario para hacer un nuevo núcleo «a medida», incluye abundante información en el directorio ***/usr/scr/linux/Documentation***, no es mala idea dar un repaso a su contenido.

Por ejemplo encontraremos en el fichero ***Changes*** una lista del software mínimo necesario para la correcta ejecución de ese kernel (esto es importante), además de pequeñas instrucciones concretas referentes a cualquier problema que pueda aparecer.

PREPARÁNDOSE PARA COMPILAR

El proceso en sí es bastante sencillo, hay una serie de comandos que se repetirán cada vez que compilemos. Esta parte es puramente mecánica y la única complicación es elegir correctamente entre la gran cantidad de opciones de hardware disponibles.

Si es la primera vez que compila, la recomendación es: hágalo cuando tenga tiempo, las primeras veces debería leer cada menú e incluso las ayudas, después los recorrerá más rápidamente, aparte del tiempo que lleva la compilación en sí.

PRIMER PASO, CONFIGURAR EL NÚCLEO

Elegir «que» va a componer nuestro nuevo núcleo. Aquí seleccionaremos que se cargará como módulo y que en la parte monolítica. Existen tres modos de hacer esta tarea,

-La más peliaguda y tediosa es en modo texto,

-En formato «menús», por los que iremos seleccionando nuestra configuración

-En modo gráfico, más agradable pero requiere el conjunto TCL/TK (es un lenguaje de programación en entorno gráfico.)

Cualquiera de estas tres opciones son equivalentes en lo que referente al resultado, la diferencia está en el interface de usuario.

Siempre nos deberemos situar en el directorio ***/usr/src/linux*** para lanzar el programa de configuración que elija. La orden:

[Crispin@Globus /linux]$ make config

lanzará el programa en modo texto. No es recomendable, por lo menos las primeras veces que intente la compilación, más que nada por el formato que usa. Le irá preguntando una a una por todas las opciones. Más cómoda es la orden

[Crispin@Globus /linux]$ make menuconfig

Veremos un menú mucho más manejable que el anterior. La opción gráfica la veremos ejecutando (desde una sesión X) la orden ***make xconfig***

RECORRER LOS MENÚS

Recorreremos todas las opciones de cada submenú, decidiendo si se incluyen o no en el núcleo y en que modo lo hacen. Desde cada submenú podemos pasar al siguiente o volver al principal.

Es buena idea recorrerlos todos ordenadamente (empezando por el principio), así nos aseguramos de no dejar una opción sin revisar. Cuando tengamos más experiencia, lo usual es ir directamente a las partes que deseamos modificar.

Dentro de cada submenú marcaremos No para no incluir esa opción, Si para incluirla en la parte monolítica o M para que se cargue como módulo. Si la opción M aparece apagada significa que ese dispositivo no admite ser cargado como módulo, hay que optar por incluirlo o no.

Se encontrará en alguna (o en muchas...) ocasión un dispositivo u opciones que no sabe exactamente para que sirven. podremos recurrir a la Ayuda e incluye una pequeña descripción y recomendaciones generales del tipo: «...si no está seguro, marque «Si» en esta opción...», cosa que se agradece.

Cada opción que seleccione para ser compilada en la parte monolítica hará un núcleo un poco más grande, exigiendo así más memoria, de manera que sea cuidadoso.

Al acabar la configuración es necesario guardar la configuración seleccionada, cosa que tendremos que hacer para compilar, guardar y salir del programa de configuración. Podremos también guardar la información de esta configuración en un fichero y cargar una configuración desde un fichero.

Esto último es interesante ya que si por algún motivo cometemos un error en la compilación y es necesario rectificar un paso, no será necesario pasar por todas las opciones, si no que, cargando el fichero guardado, podremos realizar cambios únicamente en la zona afectada.

A partir de ahora iremos viendo los mas importantes puntos de los menús de configuración del kernel:

1 .Nivel de madurez del código:

Al activar esta opción se decide si se van a aceptar las opciones que todavía están en fase de pruebas. (algunas de estas opciones son en realidad muy estables e interesantes, como por ejemplo el soporte para el sistema de archivos **reiserfs**.

2. Tipo de procesador y características

En este punto seleccionaremos el tipo de procesador que se desea compilar. Se puede deshabilitar también el numero único que se genera en los chip P/III para saber que equipo es. Hay opciones para activar MTRR que permite acelerar las operaciones con los buses PCI/AGP.

3 .Soporte para los módulos cargables

Con esta opción se permitirá que se compilen los módulos cargables, independientes de un núcleo compacto.

4. Configuración General

Estas opciones permiten una configuración general del sistema. Se aceptaran módulos de memoria mas allá del primer Gigabyte, se permitirá soporte de red y soporte de PCI. Hay un soporte de optimización de PCI que servia antiguamente para configurar los datos que la BIOS no servia.

Se puede activar System V PCI para especificar que procesos se arrancan al encender el ordenador. La opción ELF es básica para poder ejecutar los programas distribuidos que están en este formato. Se da como opción la posibilidad de actuar sobre la potencia de las placas siempre que la BIOS de esta placa tenga posibilidad.

5. Plug and Play

Esta opción activa el soporte de los elementos con capacidad plug and play.

6. Dispositivos de bloques

Con esta opción se podrá activar los dispositivos de datos por bloques tales como las disqueteras. Se pueden activar algunos puertos en paralelo con estas opciones.

7. Dispositivos RAID

Se activara en caso de tener algún dispositivo RAID. RAID es un método de manejo de discos duros (soporte para más de un disco duro).

8. Opciones de red

Se puede activar la opción de los socket con la cual se permite el acceso a la red sin utilizar ningún protocolo de red especifico. Se puede permitir el acceso bidirecional entre el núcleo y los procesos de usuario.

Se podrá activar el firewall ipchians o el moderno iptables. Aquí se tendrá que activar si se desea una red TCP / IP o no. En caso que se acepte aquí iremos colocando las políticas del protocolo IP. También se podrá elegir entre los diferentes protocolos como IPX.

9. Calidad de servicio y/o cola de prioridades

Se activa o desactiva la cola de prioridades de envío de paquetes en una cola.

10. Soporte SCSI

Es esta opción se podrán elegir entre los diferentes tipos de tarjetas SCSI que se aceptan.

11. Dispositivos de Red

En este punto se elige el hardware de red. Aquí se debe elegir entre una tarjeta del tipo ethernet, token-ring, acceso a una red ppp, slip, wan, etc.

12. Dispositivos ISDN

Se activan las redes de tipo RDSI.

13. Soporte multimedia

En estos momentos las características que se activan son las de Vídeo y Audio for Linux.

14. File system

En esta opción se eligen los tipos de ficheros que el kernel va a utilizar tales como reiserfs, ext3, ext2, JFS, FAT, VFAT, DOS, ntfs, ext, minix.

15. Console Driver

En esta opción se puede activar el frame buffer (también es EXPERIMENTAL).

16. Soporte USB

Se activara la opción de USB

17. Tarjetas de sonido

Tendremos la opción de elegir la tarjeta de sonido que se adapte a nuestro equipo.

ÚLTIMOS CONSEJOS

Como ya se ha comentado, las primeras veces que se compila, puede obtener resultados no esperados, tenga en cuenta que hay opciones que dependen de otras y quizá requiera varios intentos antes de tener su propio kernel.

Recuerde guardar su configuración al terminar y si además lo hace en un fichero aparte, podrá recuperarla en ocasiones posteriores, ahorrando tiempo en la siguiente compilación.

ORDENES DE COMPILACIÓN

Una vez seleccionadas las opciones de configuración del nuevo kernel, el proceso de compilación es significativamente más sencillo.

En primer lugar debe ejecutar (desde el directorio ***/usr/src/linux***):

[Crispin@Globus /linux]$ make dep

Esta orden lee el archivo de configuración y genera el árbol de dependencias correspondiente, seleccionando los elementos que serán compilados y los que no. A continuación debe teclear la orden:

[Crispin@Globus /linux]$ make clean

Para eliminar residuos de anteriores compilaciones, para evitar errores entre versiones. Lo siguiente es el proceso de compilación en si:

[Crispin@Globus /linux]$ make zImage

También Esta orden creará una imagen del núcleo comprimida con **gzip**, también se puede emplear, la orden: ***make bzImage***, que utilizará **bzip2**. El tamaño de la imagen es importante ya que el kernel se descomprime el los primeros Mb de la RAM y permanecerá cargado en memoria mientras el OS esté en funcionamiento.

La ejecución de estas tres ordenes generan una imagen nueva del núcleo en el directorio ***/usr/src/linux/arch/i386/boot/***

El nombre de la imagen será: **zImage** o **bzImage** dependiendo de la orden utilizada. Lo siguiente es copiar la imagen al directorio /boot:

cp /usr/src/linux/arch/i386/boot/bzImage /boot

Normalmente las imágenes usan un nombre que comienza con la cadena **vmlinuz**. Si quiere seguir este patrón, agregue un identificador propio que le permita distinguir las diferentes imágenes. Por ejemplo:

cp /usr/src/linux/arch/i386/boot/ bzImage /boot/vmlinuz.2.2.16-test

No elimine imágenes anteriores antes de haber comprobado el funcionamiento correcto del nuevo núcleo, mantenerlas no tiene efectos secundarios y le permitirá seguir entrando en el sistema en caso de errores graves.

Para tener opción en el arranque de utilizar la nueva imagen es necesario modificar la información que ***lilo*** tiene guardada en el ***mbr***. Modifique el contenido del fichero ***/etc/lilo.conf***, agregando una nueva entrada para la nueva imagen. (lo más practico es copiar integra la entrada de la imagen existente y modificar los campos **label** e **image**, con los nombres correspondientes).

¡No elimine la entrada original o no podrá entrar en el sistema en caso de algún fallo!. Ejemplo de fichero ***/etc/lilo.conf*** con la imagen original.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=10
compact

image=/boot/vmlinuz-2.2.16-5.0 #Esta es la imagen original
```

label=linux
read-only
root=/dev/hda8

Ejemplo de fichero ***/etc/lilo.conf*** con las dos imágenes.

boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=10
compact

image=/boot/vmlinuz-2.2.16-5.0 #Esta es la imagen original
label=linux
read-only
root=/dev/hda8

image=/boot/vmlinuz-2.2.16-test #Esta es la imagen nueva
label=test
read-only
root=/dev/hda8

Después de esto es necesario ejecutar ***/sbin/lilo***, para que las modificaciones sean escritas en el disco. **No olvide este paso o no se agregará su imagen al menú de arranque**.

COMPILAR LOS MÓDULOS

Antes de que poder arrancar con el nuevo *kernel*, necesitará compilar las opciones que configuró como módulo. Sencillamente ejecute la siguiente orden:

make modules

Ya solo queda una cosa por hacer, instalar los módulos. Un núcleo nuevo creará sus propios módulos, mientras que recompilar un *kernel* modificará los ya existentes, mi consejo es que haga copia de seguridad previamente.

Si la compilación falla y los módulos han sido sustituidos, es posible que pierda funcionalidad. Los módulos se almacenan en ***/lib/modules/X.X.X,*** siendo las X el número de versión del núcleo, en el ejemplo: 2.4.16. Para asegurarse de que mantiene los módulos originales ejecute:

mv /lib/modules/2.4.16 /lib/modules/2.4.16-originales

Una vez instalados los módulos del kernel que estamos compilando, se creará otro directorio ***/lib/modules/X.X.X*** (siendo las X el número de versión del núcleo nuevo). Entonces ya puede renombrar de nuevo los del núcleo anterior, de esta forma podrá arrancar con cualquiera de los dos núcleos ya que cada uno identificará sus propios módulos.

Después de haber hecho la copia de seguridad de los módulos puede ejecutar la orden para instalar los módulos:

make modules_install

En este momento puede rearrancar el ordenador y en el *prompt* de Lilo seleccionar el núcleo nuevo. En caso de error, y si siguió las indicaciones de copias de seguridad, podrá arrancar con el antiguo y realizar las modificaciones necesarias.

Si arranca con el kernel recién compilado, dé un repaso por el sistema para comprobar que todo funciona como debería, disquetera, lector de CD-Rom, soporte de red, sistemas de archivos, puertos serie y paralelo, etc.

PARCHES PARA EL KERNEL

Un parche para el kernel no es mas, que un fichero que solamente contiene información, sobre las lineas de código que han cambiado desde la versión precedente del núcleo. De esta manera, solamente te tienes que bajar un fichero con los cambios, en vez del núcleo al completo. El ahorro en cantidad de Mb bajados es bastante considerable, sobre todo para aquellos que dependen del modem y no tiene una conexión buena a internet .

Algo muy a tener en cuenta si vais a actualizar el núcleo por medio de parches, en vez de bajaras el núcleo al completo, es que tenéis que ir actualizando de versión a versión. Para que se entienda un poco mejor, aquí tenéis un ejemplo:

Si tenéis el núcleo 2.4.0 y vais a actualizarlo al 2.4.1, os podéis bajar el fichero pacta-2.4.1.gz [70Kb] en vez, del núcleo 2.4.1 al completo [12.5Mb]. Pero si tenéis el núcleo 2.4.0 y vais a actualizar al 2.4.4, NO os vale bajaras el fichero pacta-2.4.4.gz nada mas, tendríais que bajaros el 2.4.1, 2.4.2, 2.4.3 y 2.4.4. Esto es porque los ficheros patch solamente contienen los cambios de versión a versión.

Si la diferencia entre la versión que tenéis y la que queréis instalar, es muy grande (p.ej: del 2.2.0 al 2.2.35), no os merece la pena actualizar por medio de parches, en este caso bajaros la versión completa.

Una vez bajado el fichero patch (se pueden bajar del mismo subdirectorio donde esta la versión completa), que necesitas para actualizar el kernel, hay que aplicarlo al núcleo que tienes y compilar de nuevo. El procedimiento para actualizar el núcleo por medio de ficheros patch es el siguiente:

1. .Copia el fichero patch-XX.YY.ZZ.gz al directorio /usr/src :

cp patch-XX.YY.ZZ.gz /usr/src/

2.Cambia a este subdirectorio y descomprime el fichero: gunzip patch-XX.YY.ZZ.gz

3.Aplica el parche: patch -s -p0 < patch-XX.YY.ZZ

4.La opción -s hará que solo se impriman mensajes de error. Si no recibes ningún mensaje de error (como debería de ser ;-)) solamente te queda entrar en /usr/src/linux:

cd /usr/src/linux

5.Y ejecutar make clean, make xconfig, make dep, make bzImage.

NOMBRES DE LOS DISPOSITIVOS

Hay que tener en cuenta que al compilar nuestro kernel, una de las principales cosas que vamos a hacer es generar modulos para que algunos componentes hardware funcionen. Hay nuchos componentes hardware que se tienen que conectar a dispositivos. Estos dipositivos se encuentran situados en el directorio /dev. En este directorio todos los archivos tendran dos números. Uno es el mayor y el minor y estos indican a que objeto hardware va a referencia ese dipositivo.

Asi, una serie de dispositivos con sus nombres, para tener una referencia de lo que es cada uno:

-sd? -- Disco SCSI

-hd? -- Disco duro físico

-fd? -- Disqueteras

-ttyS? -- Puertos serie

-tty? -- Consolas

CONFIGURACIÓN Y CARGA DE MÓDULOS

El núcleo del sistema es un programa grande y complejo que funciona en el modo privilegiado del procesador, por lo cual puede leer y escribir en los puertos y tiene acceso a toda la memoria.

Dado que disponemos del código fuente del núcleo, la máxima flexibilidad imaginable en cuanto a sus características y configuración se alcanza modificando ese código y recompilándolo.

Para evitar tener que recompilar cada vez que se cambia el hardware o se necesita añadir o eliminar alguna funcionalidad, algunas partes del código objeto no se enlazan con el resto del núcleo en tiempo de compilación: son los módulos.

Típicamente, pero no en todos los casos, se compilan como módulos los manejadores de dispositivo (device drivers). Cuándo enlazar directamente en el núcleo:

1.Cuando el dispositivo o servicio se usa en el arranque (ide, scsi, etc.)

2.Cuando el dispositivo o servicio es de uso frecuente o constante (tarjeta de red, sistemas de ficheros, etc).

Cuándo compilar como módulo:

1.Cuando el dispositivo o servicio se usa de tarde en tarde (escáner, puerto paralelo, etc.)

2.Cuando el dispositivo puede variar sus características (dispositivo externo scsi, discos reemplazables en caliente, etc.)

3.Manejadores no incluidos en la versión oficial del núcleo (experimentales, locales, especiales, etc.)

La administración de los módulos a nivel básico es muy sencilla, limitándose a los procesos de inserción, consulta de los instalados y extracción. Lógicamente, sólo el superusuario puede administrar directamente el núcleo y los módulos. Para saber qué módulos hay instalados se usa la orden lsmod . Prueba:

lsmod

Para instalar un módulo se usa **insmod** o **modprobe** . Este último comprueba dependencias entre módulos e inserta todos los necesarios, no sólo el que le pasamos como argumento. El programa **depmod** es el que calcula las dependencias y las guarda en el fichero

/lib/modules/$(uname -r)/modules.dep.

Precisamente, los módulos se encuentran siempre en directorios con el nombre de versión del núcleo, a partir de /lib/modules. Compruébalo en tu sistema.

insmod lp

modprobe -k lp

lsmod

Para extraer un módulo usaremos la orden **rmmod** y **modprobe.**

modprobe -r lp

rmmod lp

lsmod

La inserción y extracción de módulos se encuentra normalmente automatizada mediante un hilo de ejecución del núcleo, **kmod**, que llama a modprobe cuando un proceso le pide abrir un dispositivo que no conoce o un servicio del que no dispone. Por ejemplo:

Cuando se pide arrancar un servicio de red sobre ethernet y no existe el dispositivo instalado, kmod ejecuta modprobe eth0. Cuando se pide montar un disco scsi y no está el manejador de la tarjeta scsi instalado, kmod ejecuta modprobe scsi_hostadapter.

Si un proceso intenta abrir un fichero especial de dispositivo de bloques con "major number" M y "minor number" m, kmod ejecuta modprobe block-major-M-n. Modprobe hace uso del fichero /etc/modules.conf, que conviene conocer, y que le permite traducir estos nombres simbólicos a nombres de módulos. Examina este fichero en tu sistema. Te darás cuenta de que en esta distribución ese fichero no se modifica a mano, sino que la utilidad update_modules lo genera a partir de los ficheros del directorio /etc/modutils .

Es en este directorio donde deberemos añadir los alias necesarios para que **modprobe** instale los módulos especiales que pudiéramos necesitar en alguna ocasión.

CREACIÓN DEL ALIAS (FIJAR MÓDULOS)

Se puede crear a la vez un alias para cada módulo con lo que la llamada a este llamara a los módulos.

Así, si miramos en el archivo ***/etc/modules.conf*** tendremos todos los módulos con todos los alias. En el caso de la red tendremos una entrada como:

alias eth0 ne2k-pci

Hay que tener en cuenta las versiones del kernel para poder ver este archivo. Así en versiones anteriores a 2.2.x el archivo de configuración es el /***etc/conf.modules***.

Hay otro archivo de configuración de los alias. Este archivo es ***/etc/modutils/aliases***. Este archivo es utilizado por ***update-modules*** para comprobar las dependencias y actualizar los módulos a través del archivo /***etc/modules.conf***.

Cuando se quiere cargar un módulo, este para comprobar las dependencias tiene que mirar el archivo ***/lib/modules/versión_del_kernel/modules.dep*** que será:

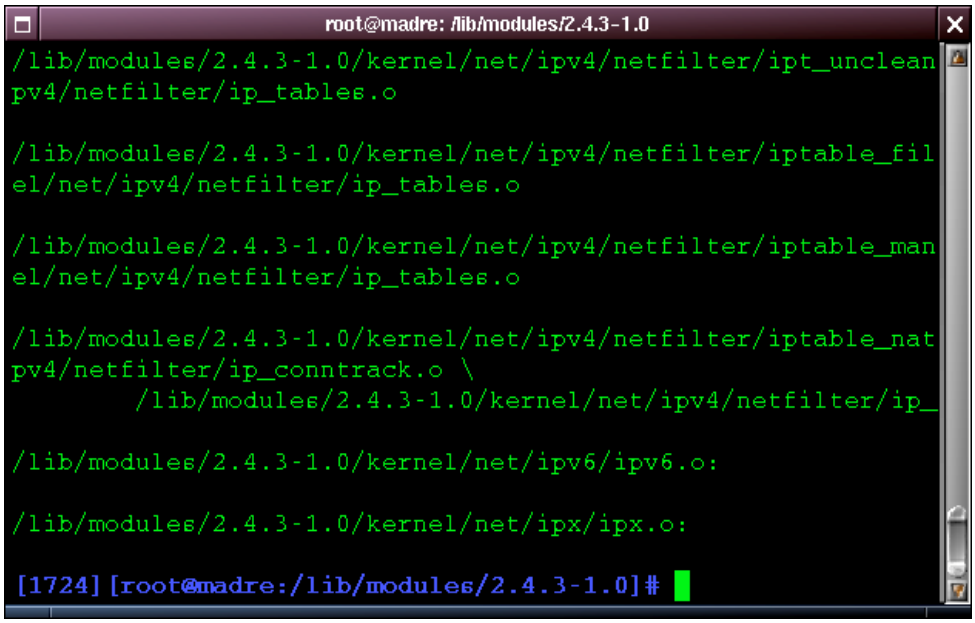
Por lo tanto, cuando necesitemos fijar un módulo, para que este sea cargado al arranque de la máquina, debemos editar el fichero ***/etc/modutils/aliases***, incluyendo en el nombre del módulo que queremos fijar. Es necesario hacerlo con la sintaxis propia del fichero, que es algo parecido a esto (módulos del interface ppp):

alias char-major-108 ppp_generic

alias /dev/ppp ppp_generic

alias ppp-compress-21 bsd_comp

A continuación ejecutaremos el comando ***update-modules*** para que sea esta orden la que escriba el alias en el fichero: /***etc/modules.conf***, actualizándolo. No es adecuado editar directamente el /***etc/modules.conf.***


root@madre: /lib/modules/2.4.3-1.0
/lib/modules/2.4.3-1.0/kernel/net/ipv4/netfilter/ipt_unclean
pv4/netfilter/ip_tables.o
/lib/modules/2.4.3-1.0/kernel/net/ipv4/netfilter/iptable_fil
el/net/ipv4/netfilter/ip_tables.o
/lib/modules/2.4.3-1.0/kernel/net/ipv4/netfilter/iptable_man
el/net/ipv4/netfilter/ip_tables.o
/lib/modules/2.4.3-1.0/kernel/net/ipv4/netfilter/iptable_nat
pv4/netfilter/ip_conntrack.o \
/lib/modules/2.4.3-1.0/kernel/net/ipv4/netfilter/ip_
/lib/modules/2.4.3-1.0/kernel/net/ipv6/ipv6.o:
/lib/modules/2.4.3-1.0/kernel/net/ipx/ipx.o:
[1724][root@madre:/lib/modules/2.4.3-1.0]#

modules.dep

SERVICIOS DEL SISTEMA

Cuando arranca el sistema, el último paso es el arranque de los servicios. Éstos sirven para proporcionar una serie de funcionalidades al sistema. Mediante estos servicios nuestra máquina será capaz de realizar múltiples operaciones que podemos dividir en dos clases:

-**Servicios de red**: Son aquellos que permiten a nuestra máquina ofrecer servicios a otras conectadas a la red. Algunos ejemplo de éstos son servir páginas web y bases de datos, permitir la administración remota, importar directorios a otras máquinas, compartir ficheros con redes basadas en windows, etc.

- **Servicios locales**: Son más reducidos y ofrecen funcionalidades muy concretas. Ejemplos de servicios locales son gestor avanzado de energía, cursor del ratón para el shell, ejecución de tareas planificadas...

Todos estos servicios se encuentran almacenados en un único directorio, el /etc/rc.d/init.d/, pero para que sean ejecutados al arrancar el sistema depende de otros directorios nombrados con /etc/rcX.d/ siendo X el nivel de ejecución.

NIVELES DE EJECUCIÓN

Explicaremos el concepto de nivel de ejecución: Dentro de la estructura ***/etc/*** hay un fichero llamado inittab en el que se encuentran las definiciones de los diferentes niveles de ejecución (runlevels) e indica también cuál será el runlevel por defecto al arrancar.

La línea ***id:3:initdefault:*** de este fichero indica que se iniciará el runlevel 3. A continuación describimos cada uno de los posibles niveles de ejecución:

- **0:** halt: Sistema apagado, ningún servicio en ejecución

- **1**: Single user mode: Modo monousuario. No tendremos más que una terminal virtual y no dispondremos de acceso a los recursos de red. Se cargan los servicios mínimos.

- **2:** Multiuser

- **3**: Full multiuser mode: Tanto el 2 como el 3 proporcionan servicios de red y acceso completo a la máquina. El runlevel 3 es el nivel de ejecución por defecto.

- **4:** unused: Este es un nivel de ejecución de usuario, es decir, no está definido en ningún estándar cuál debe ser su contenido ni los servicios que debe lanzar o detener.

- **5:** X11: Este nivel de ejecución es igual que el 3 con la excepción de que también arrancará el servicio de login gráfico, de forma que se nos presentará el sistema cd X Window en lugar del clásico "Login: "

- **6:** reboot: Este nivel de ejecución para todos los servicios y reinicia el ordenador. Cada enlace de los directorios /etc/rcX.d/ lleva el siguiente formato:

[S|K] nn Nombre_servicio. La S indica que el servicio especificado se debe arrancar (Start), mientras que una K indica que dicho servicio debe ser detenido (Kill). nn es un número comprendido entre 00 y 99 que indica el orden de ejecución de los servicios.

00 se ejecutará antes que todos los demás. Esto es útil por ejemplo para iniciar antes la red que los servicios que van a correr sobre ella. Por último en nombre del servicio es el mismo que tendrá el script al que se enlaza e indica su función.

Como ejemplos tendremos los siguientes:

S10network: Inicia los interfaces de red con una prioridad 10

K75netfs: Finaliza la ejecución de los sistemas de ficheros exportados por red.

Realmente estos valores son parte del estándar de niveles de ejecución y su comportamiento depende de los enlaces que se encuentren en su directorio de nivel de ejecución y no del número. Por ejemplo, el nivel 6 es el de reinicio porque dentro del directorio /etc/rc6.d/ se encuentran enlaces que indican la finalización de todos los procesos y una llamada a reboot que provoca el reseteo de la máquina. Si eliminamos este enlace, ésta no se reiniciará.

Los enlaces llevan al directorio /etc/init.d/ de forma que para cada nivel de ejecución se crean enlaces simbólico a los script de este directorio.

INICIO MANUAL DE SERVICIOS

Normalmente, todos estos scripts son llamadas a los verdaderos programas que servirán información y su única función es activarlos y desactivarlos con los parámetros adecuados.

Un ejemplo podría ser el servidor de páginas web Apache. Éste se podría iniciar llamando directamente al programa con la orden

/usr/sbin/httpd

Para finalizar su ejecución, deberíamos matar al proceso, bien con 'killall httpd' o bien con 'kill nnn', siendo n su pid. Para simplificar todos estos comandos, los scripts de ***/etc/rc.d/init.d*** incluyen por lo general tres opciones a modo de parámetros:

- **start**: Hace que el servicio se active sólo si no estaba ya en ejecución

- **stop**: Para el servicio.

- **restart**: Para el servicio y a continuación lo vuelve a iniciar.

Aparte de facilitar el arranque/parada de los servicios, éstos scripts llevan un control de los servicios que han sido lanzados, de forma que algunos se implementan el parámetro 'stat' que nos devuelve el estado del servicio deseado.

Ejemplo inicio servidor ftp:

madre@madre:/# /etc/init.d/proftpd start
Starting proftpd: Allowing sessions again [SI]

Si el servicio ya estaba iniciado nos alertará de ello y mostrará [NO] indicando que no se realizó con éxito. También mostrará este indicativo
cuando se produzca algún error en la operación.

Para ver las opciones que admite cada servicio debemos llamar al script sin ningún parámetro. Esto nos devolverá su uso sintáctico:

Uso: /etc/init.d/httpd {start|stop|restart|reload|condrestart|status}

Los valores que se encuentran entre los corchetes son cada uno de los parámetros que admite el servicio pudiendo elegir entre UNO de ellos.

Para comprender mejor el funcionamiento del arranque de los servicios lo representaremos en forma de esquema

1.-Arranque de sistema (Pasa de estado N a estado 3)

2.- Lee los ficheros Knn del directorio /etc/rc3.d que enlazan a /etc/rc.d/init.d/servicio pero con el argumento stop. (Esto no lo realiza en el arranque ya que no hay ningún servicio ejecutado)

3.- Igual que el paso anterior pero ejecuta todos aquellos enlaces que sean de inicio (Snn) y les pasa el parámetro start

Podemos cambiar el nivel de ejecución de nuestro sistema mediante el comando 'telinit n' donde n es el nivel al que se desea cambiar.

SERVICIOS INETD

Uno de los servicios más importantes del sistema es el inetd, ya que se encarga de ejecutar sus propios servicios, pero en este caso orientados exclusivamente a internet. Podemos hacer una diferenciación entre los servidores que son lanzados por inetd y los que son activados al iniciar un determinado nivel de ejecución.

Aquello que se lanzan desde inetd formará parte de un grupo de servicios que se inician o detienen a la vez, no es posible arrancar uno sólo. Comúnmente desde inetd se lanzan los servicios de internet más comunes, pero no tiene porqué ser así.

Pondremos el ejemplo de un servidor FTP. Al arrancar el servicio inetd, éste lee el fichero /etc/inetd.conf y lanza un demonio para cada una de los servicios que incluya. En el caso del ftp lanzaría el fichero in.ftpd mediante tcpd y quedaría en ejecución el servidor ftp. Si paramos el servicio inetd, finalizarán todos aquellos servicios que tuviese asociados.

Podemos ejecutar también los servicios de internet en modo standalone (que traducido viene a ser "en solitario"). De esta forma, un servidor ftp tendría su propio script de ejecución en el directorio /etc/rc.d/init.d/ y podría ser lanzado y parado independientemente de los demás servicios.

Se recomienda lanzar todos los servicios posibles en modo standalone, tanto por seguridad como por versatilidad.

Este es un ejemplo de cómo lanzar el mismo servicio desde inetd y en modo standalone:

inetd.conf: smtp stream tcp nowait mail /usr/sbin/exim exim -bs

standalone (comando): /etc/rc.d/init.d/exim

Ambos métodos llaman al mismo demonio, pero en el primer caso, exim (el programa encargado de gestionar el correo) es dependiente de inetd.

Como hemos comprobado, las líneas del fichero /etc/inetd.conf tienen varios parámetros, los analizaremos para comprender mejor su funcionamiento:

Cada una de las líneas de este fichero está dividido en 7 campos:

- <**nombre_servicio**>: Indica el nombre del servicio que se va a lanzar. Puede obtener una descripción de los well-known-services en el fichero /etc/services que describiremos más tarde.

- <**tipo_socket**>: Este campo indica el tipo de socket que se usará en las conexiones de este servicio. Pueden ser de tipo stream, datagrama, raw, rdm o seqpacket.

- <**protocolo**> Aquí se indica el protocolo que se utilizará, así como el número de puerto en el que el demonio estará escuchando.

-<**flags**> Este campo puede contener dos valores: wait y nowait y deberemos usar nowait con todos los tipos de socket excepto con dgram que admite los dos valores. Únicamente indicará si el servidor puede crear o no su propio socket cuando se le solicite información.

Si indicamos Wait sólo podremos abrir ese socket simultáneamente ya que queda bloqueado el servicio hasta que se cierre dicho socket.

- <**usuario**> Todo proceso en Linux es ejecutado por algún usuario, con este campo indicamos qué permisos va a tener el servicio al ejecutarse. Tendrá los mismos permisos que el usuario que lo ejecutó. Debemos evitar la ejecución de demonios con usuario root.

- <**ruta_servidor**> Es el servidor que se iniciará para atender las peticiones. Aunque en este campo debería figurar el nombre del demonio, normalmente, tenemos un programa "filtro", un TCP Wrapper, que comprueba permisos y decide si se tiene acceso a un determinado servicio. El tcp wrapper es el fichero ***/usr/sbin/tcpd.***

- <***parámetros***> Como parámetros para el tcpd le pasaremos el nombre del programa demonio que atenderá las peticiones, aquí debemos introducir el nombre del fichero, así como los posibles parámetros que necesite para funcionar.

EL FICHERO /ETC/SERVICES

En este fichero se guarda una relación de los servicios y sus puertos asociados más comunes. Guarda una estructura de tres campos más un comentario explicativo de la función de dicho servicio.

El primer campo es el nombre "amistoso" para el servicio que será utilizado para denominarlo o buscarlo. Ej: En los ficheros de log aparecerá el nombre especificado en este campo si alguien intenta una conexión a su puerto asociado.

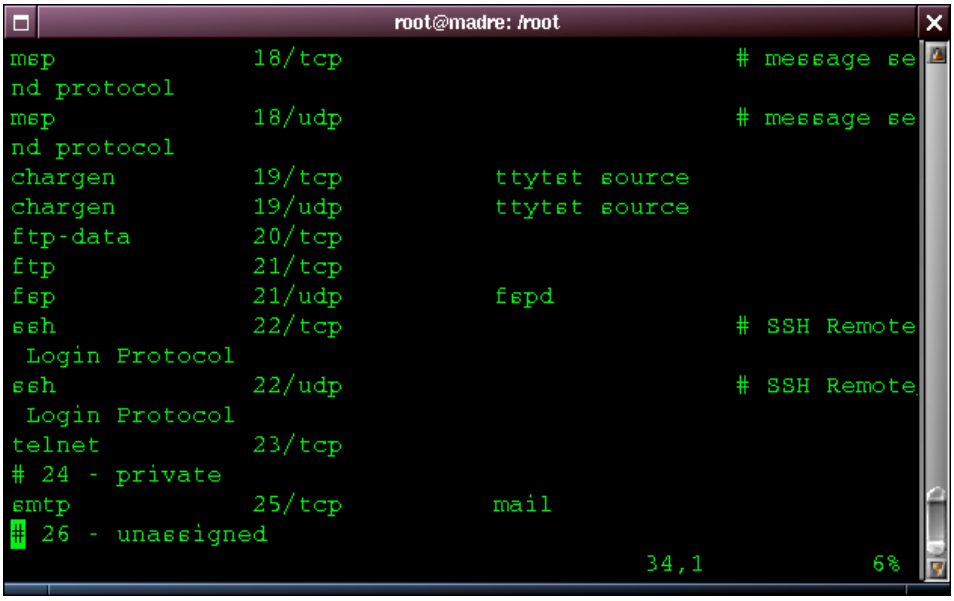
El campo de puerto/protocolo indica el número de puerto que se utilizará para este servicio. El protocolo indica el tipo de conexión que se realizará en este servicio.

Los valores comunes son tcp y udp. Muchos servicios tienen asociados dos puertos con el mismo número: Uno con UDP y otro con TCP. Esto es normal y varios demonios lo requieren.

- **Alias**: En este campo puede especificar uno o varios alias o nombres opcionales para este servicio. Cada alias debe estar separado por un tabulador. Puede encontrarle utilidad a este campo para recordar nombres de servicios complicados.

netbios-ssn -> samba

Estos son algunos de los servicios más comunes:



[Servicio] [Puerto/proto] [alias] [Comentario / Explicación]

PROGRAMAS UTILITARIOS

Para poder sacar provecho a los conceptos que hemos aprendido tenemos varias aplicaciones.

FUSER

Esta aplicación nos muestra los **PID** de los procesos especificados, ya bien sean un archivo o un servicio. Además podremos matar los servicios que deseemos o bien mostrar que usuarios esta utilizando ese servicio.

Asi los parámetros más comunes que utilizaremos serán los siguientes:

-**a:** Muestra todos los archivos especificados en la linea de comandos.

-k: Mata los procesos señalados.

-**l**: Lista todas las señales conocidas.

-**u**: Muestra el usuario especificado a cada servicio.

Asi si lo que queremos es mostrar todas las señales que se aceptan escribiremos:

fuser -u

Si lo que queremos es matar el telnet tendremos que escribir:

fuser -k telnet/tcp

Si lo que queremos es ver que usuario esta utilizando el webmin tendremos que poner:

fuser -u webmin/tcp

LINUXCONF

La utilidad linuxconf utiliza una serie de archivos y de scripts para configurar el sistema. Así cada vez que cambiemos la configuración de nuestro equipo, todos los datos se guardarán en los archivos de configuración del programa en cuestión. En el caso de las redes los más importantes son: /etc/resolv.conf, /etc/hosts, /etc/host.conf, /etc/hostname y los contenidos en el directorio: /etc/network, sobre todo el archivo, interfaces.

Es evidente que para que una red pueda levantarse (ya sea durante el arranque de la máquina o a voluntad), el kernel tiene que conocer la tarjeta de red. Si el núcleo contiene la información necesaria para controlar un determinado modelo de tarjeta, se encargará de hacerlo. De lo contrario habrá que instalar el módulo específico que estará en: /lib/mdules/2.4.18../kernel/drivers/net, usando el comando modprobe.

Si el módulo no se encuentra en el directorio mencionado arriba, será necesario compilar el kernel y habilitar los módulos necesarios.

Estos archivos se pueden modificar manualmente, los que no tienen demasiada experiencia en manipular ficheros de configuración, quizá prefieran utilizar el programa linuxconf.

WEBMIN

Webmin es un administrador general del sistema a través de un browser (**navegador**) . Con esta aplicación se puede administrar los diferentes puntos de administración de un sistema .

La cualidad principal es que ***WEBMIN*** lo asociamos a un puerto y a través de cualquier browser y desde cualquier punto de la red podremos administrar la máquina .

Cuando se entra en webmin se hará a través de una dirección **IP** y una dirección de puerto, así una llamada desde un browser sería:

http://192.168.10.43:10000

La administración se divide en cinco puntos, los cuales iremos desglosando :

CONFIGURACIÓN GENERAL DE WEBMIN

1. **Configuración del WEBMIN**: En este apartado podremos elegir el puerto en el cual escucharemos, los usuarios que tendrán acceso, el idioma en el que estará traducido, modo de autenticación, formato del escritorio, sistema operativo a utilizar, etc. Como se podrá ver son muchísimas las características a poder modificar.

2. **Historiales de acción**: En este apartado buscaremos los diferentes archivos de historiales que utilicemos.

3. **Usuarios de WEBMIN**: En este apartado administraremos tanto los usuarios de webmin como los módulos modificables que existan (En este caso se decidirá que acción se puede ejecutar en cada módulo).

4. **Servidores WEBMIN**: En este apartado se podrán buscar otros servidores ***WEBMIN*** situados en una red.

SISTEMA

En este apartado de **webmin** se podrán administrar los siguientes puntos:

Arranque y parada: aquí configuraremos todos procesos del sistema en el arranque y en la parada.

Configuración Init de Sys V: aquí, en cada nivel de ejecución podremos ir añadiendo o eliminando servicios

Cuotas de disco: aquí se distribuirá la cantidad de espacio que un usuario podrá tener en un sistema de cuotas

Exportación de NFS: Nos permitirá exportar sistemas de ficheros de red.

Historiales de sistema: aquí podremos activar, desactivar o ver los archivos log del sistema.

Autenticación PAM: Activaremos los servicios de autenticación de PAM

Paquetes de Software: Podremos instalar, desinstalar o actualizar los paquetes.

Procesos en curso: Se podrá actuar sobre todos los procesos que en ese momento estén en curso.

Paginas del manual: Es un buscador de páginas de ayuda

Sistema de archivo de Disco y red: Podremos montar y desmontar las diferentes particiones del sistema.

Tareas planificadas de CRON: Trabajaremos con el programa ***cron*** para ejecutar tareas programadas temporalmente.

Usuarios y grupos: Administrador de grupos y usuarios

SERVIDORES

En esta sección podremos configurar todas las opciones de nuestro servidor. Las opciones de la versión 0.75 son las siguientes:

Configuración de archivos de windows mediante Samba: aquí se podrá configurar el servidor samba para interconectar una red Microsoft con una red Linux

Configuración de Postfix: Configuración de MTA (Mail Transport Agent), con el cual conseguiremos la salida de correo.

Configuración de sendmail: Configuración del administrador de correo.

Configuración de listas de majordomo: Instalación de las listas de correo.

Nombres y claves de acceso de usuarios PPP: Podremos configurar nuestro modem y el acceso de usuarios a este.

Servicios y protocolos de internet: Se podrá configurar que protocolos queremos usar en nuestra máquina.

Servidor FTP: Configuración del servidor FTP.

Servidor PROXY squid: aquí podremos configurar nuestro servidor de PROXY para que diferentes máquinas se puedan conectar a internet mediante una sola salida.

Servidor WEB Apache: Podremos configurar todos los módulos que se podrán instalar en el servidor WEB.

Servidor de base de datos MySQL: Configuración del servidor de bases de datos, en el cual podremos hacer consultas y modificaciones de bases de datos que podremos crear.

Servidor DHCP: Configuración del servidor dinámico de direcciones IP.

Servidor de DNS BIND 8: Configuración del servidor de Nombres en su versión 8.

Servidor de bases de datos PostgreSQL: Al igual que con MySQL podremos crear y modificar bases de datos y modificar las existentes.

HARDWARE

En este punto podremos configurar los dispositivos hardware que conectemos a nuestro equipo.

Administración de impresoras: En este punto configuraremos todas las impresoras que se utilicen en la red.

Configuración de red: aquí podremos configurar desde nuestro interfaz de red, nuestras direcciones de enrutamiento, el cliente DNS y la dirección de la máquina.

Configuración de arranque de Linux: Configuraremos el LILO en la máquina en la que actuemos.

Hora del sistema: Podremos sincronizar nuestras máquinas con las horas del hardware y cambiar las fechas del sistema.

Particiones en discos locales: Podremos particionar nuestros discos cambiando el tamaño de la existentes o creando unas nuevas.

RAID de Linux: Se configurará el servicio de tablas de discos en nuestro sistema.

OTROS

Configuraremos otras características que no se engloban en ninguno de los puntos anteriores.

Crear comandos: Esta opción nos permitirá crear nuevos comandos para utilizarlos dentro de nuestro sistema.

Login mediante telnet: Podremos realizar una conexión telnet a nuestro equipo.

Estado del sistema y del servidor: En esta sección se comenta que opciones se están ejecutando de las configuradas en la pestaña de servidor.

Manejador de archivos: Nos aparecerá un programa de administración de archivos de nuestro servidor.

Módulos de Perl: Podremos instalar los módulos de perl a través del CPAN.

CONTROL DE TAREAS Y PROCESOS

El comando **ps** sin opciones visualiza la lista de procesos que se están ejecutando, en el terminal actual. La misma orden con las opciones **-ef** nos da mucha más información.

PRIMER PLANO Y SEGUNDO PLANO

Un proceso puede estar en Primer plano **fg** (foreground) o en Segundo plano **bg** (background). Solo puede haber un proceso en primer plano al mismo tiempo (en cada consola), el proceso que esta en primer plano, es el que interactúa con nosotros (en los entornos gráficos puede haber varios procesos en primer plano y en ese caso interactuamos con aquel que *tiene el foco)*, recibe entradas de teclado, y envía las salidas al monitor. (Salvo, por supuesto, que haya redirigido la entrada o la salida). El proceso en segundo plano, no recibe ninguna señal desde el teclado por lo general, se ejecutan en silencio sin necesidad de interacción.

Los procesos pueden ser suspendidos con la combinación **Ctrl+Z**. Un proceso suspendido es aquel que no se esta ejecutando actualmente, sino que esta temporalmente parado. Después de suspender una tarea, puede indicar a la misma que continúe, en primer plano o en segundo, según necesite. Retomar una tarea suspendida no cambia en nada el estado de la misma la tarea continuará ejecutándose justo donde se dejó. Tenga en cuenta que suspender un trabajo no es lo mismo que interrumpirlo. Cuando usted interrumpe un proceso (generalmente con la pulsación de <**Ctrl+C**>), el proceso muere, y deja de estar en memoria y utilizar recursos del ordenador.

Una vez eliminado, el proceso no puede continuar ejecutándose, y deberá ser lanzado otra vez para volver a realizar sus tareas. También se puede dar el caso de que algunos programas capturan la interrupción, de modo que pulsando <**Ctrl+C**>no se para inmediatamente. De hecho, algunos programas simplemente no se dejan matar por ninguna interrupción.

Ctrl+C tiene el mismo efecto sobre el proceso que se está ejecutando en esa consola que una orden **kill -2 [nº de proceso]**

ENVÍO A SEGUNDO PLANO Y ELIMINACIÓN DE PROCESOS

Una forma de mandar procesos a segundo plano es añadiendo un carácter & al final de cada comando. Como se ver en la *figura 4* hemos lanzado cuatro procesos **yes** en segundo plano, la shell le asigna a cada uno un número de tarea [1] y el kernel un número de proceso **PID** (cualquiera de estos dos número pueden utilizarse para manejar el proceso). Para chequear el estado del proceso, utilice el comando interno de la shell **jobs**:

Para eliminar una tarea, utilice el comando **kill**. Este comando toma como argumento un numero de tarea o el nombre del comando (siempre precedidos por el símbolo **%**), o el numero **PID** del proceso. En la *figura 4* vemos las diversas formas de matar el proceso yes.

Primero, con el comando **jobs** comprobamos los números de tarea de los cuatro procesos yes lanzados. Matamos el primero de ellos con el comando kill y como argumento el PID, el segundo lo acabamos dándole como argumento a kill el número de tarea, a continuación con el comando **fg 3** traemos a primer plano (foreground), la tarea 3 y una vez en primer plano la terminamos con la combinación Ctrl+C. Tecleamos la orden jobs y vemos que ya sólo nos queda un proceso yes ejecutándose, por lo tanto podemos matarlo usando su nombre (yes) como argumento del comando kill.

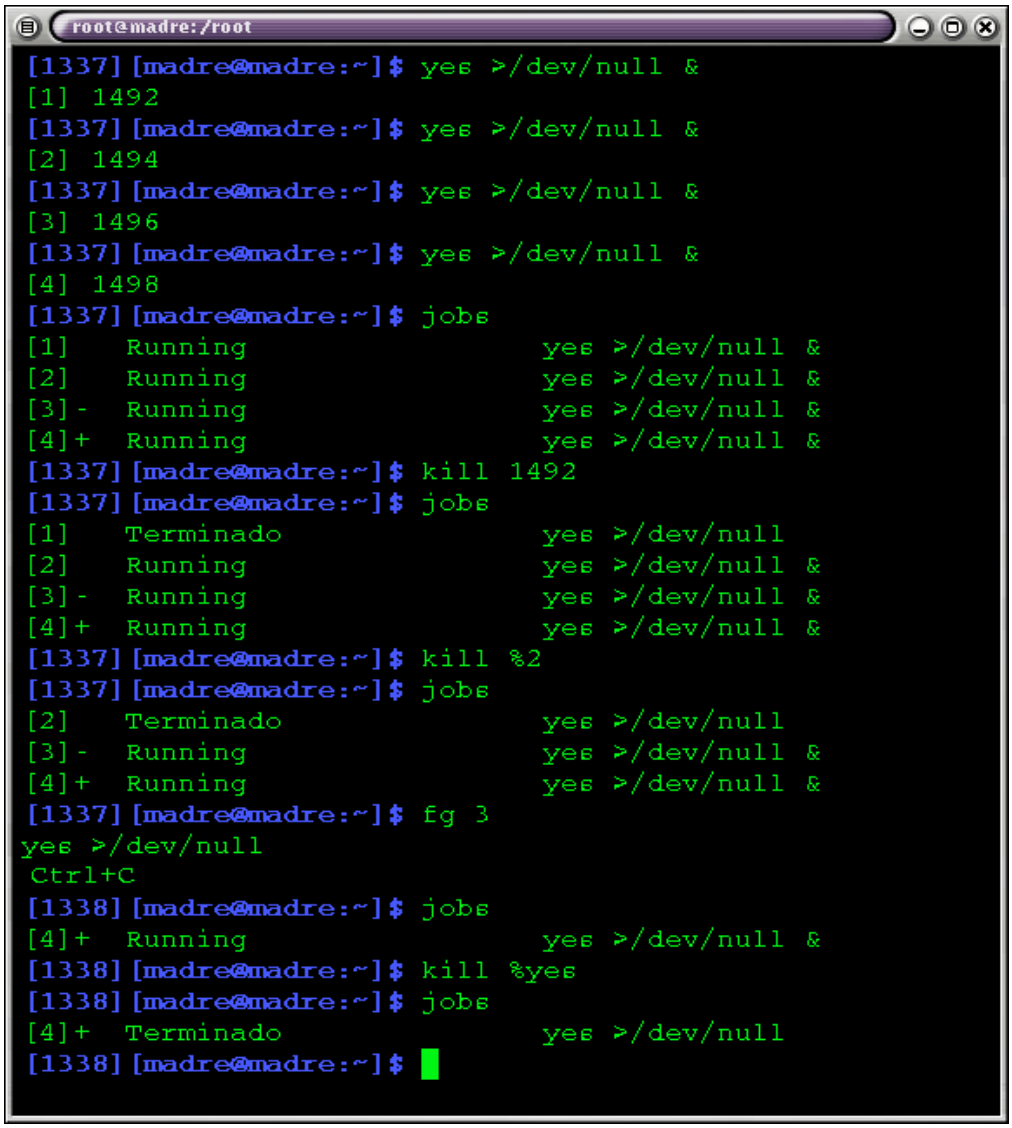


figura 4

PARADA Y RELANZAMIENTO DE TAREAS

Podemos lanzar un proceso normal (en primer plano), pararlo, y después relanzarlo en segundo plano. Primero, lanzamos el proceso en primer plano como se haría normalmente.

Ahora, en vez de interrumpir la tarea con <Ctrl+C>, suspenderemos la tarea. El suspender una tarea no la mata: solamente la detiene temporalmente hasta que Ud. la retoma. Para hacer esto usted debe pulsar la tecla de suspender, que suele ser <Ctrl+Z>.

Mientras el proceso esta suspendido, simplemente no se esta ejecutando. No gasta tiempo de CPU en la tarea. Sin embargo, usted puede retomar el proceso de nuevo como si nada hubiera pasado. Continuara ejecutándose donde se dejo. Para relanzar la tarea en primer plano, use el comando **fg** .

Pare la tarea de nuevo, con <Ctrl+Z>. Esta vez utilice el comando **bg** para poner la tarea en segundo plano. Esto hará que el comando siga ejecutándose igual que si lo hubiese hecho desde el principio con **&** como en la sección anterior.

Hay una gran diferencia entre una tarea que se encuentra en segundo plano y una que se encuentra detenida. Una tarea detenida es una tarea que no se esta ejecutando, es decir, que no usa tiempo de CPU, y que no esta haciendo ningún trabajo (la tarea aun ocupa un lugar en memoria) .

Una tarea en segundo plano, se esta ejecutando, y usando memoria, a la vez que completando alguna acción mientras usted hace otro trabajo. Sin embargo, una tarea en segundo plano puede intentar mostrar texto en su terminal, lo que puede resultar molesto si esta intentando hacer otra cosa. Por ejemplo, si usamos el comando **yes &** sin redirigir stdout a /dev/null, una cadena de **yes** se mostraran en su monitor, sin modo alguno de interrumpirlo (no puede hacer uso de <Ctrl+C>para interrumpir tareas en segundo plano). Para poder parar esos interminables yes, tendría que usar el comando **fg** para pasar la tarea a primer plano, y entonces usar <**Ctrl+C**>para matarla.

Otra observación. Normalmente, los comandos fg y bg actúan sobre el ultimo proceso en segundo plano (indicado por un «+» junto al numero de tarea cuando usa el comando jobs). Si usted tiene varios procesos corriendo a la vez, podrá mandar a primer o segundo plano una tarea especifica indicando el *ID de tarea* como argumento de fg o bg. No se pueden usar los *ID de proceso* con fg o bg.

VARIABLES

Una variable es una pequeña porción de memoria a la que se le asigna un valor (puede ser un número, una ruta de directorios..., etc.) que se puede leer o consultar e incluso modificar. Hay tres tipos de variables:

- de **entorno**, que forman parte del entorno del sistema. (Pueden ser locales o globales).

- **incorporadas**, que las proporciona el sistema y no pueden ser modificadas con un programa *shell.*

de **usuario**, modificables con la *shell.*

Cuando empezamos una sesión se ejecuta la shell que esté definida para el usuario en el archivo ***/etc/passwd*** (normalmente **bash**), el intérprete de comandos carga varios archivos que localiza en diferentes directorios del sistema. Estos archivos contienen entre otras cosas, variables de entorno que definen los valores del intérprete de comandos y de otros programas. Bash tanto al inicio de la sesión como a lo largo de ella lee estos archivos: **/home/usuario/ .bashrc**, **bash_profile**, **bash_history** y **/etc/profile**, **/etc/bashrc .** El interprete de comandos bash reconoce alrededor de 60 variables de entorno diferentes y permite que el usuario defina otras personalizadas.

Un ejemplo de variable es **PATH**, define las rutas que deben ser "miradas" por la shell para buscar comandos o ejecutables, otro sería la variable **EDITOR**, que define el editor por defecto que se abriría para editar un archivo y también está la variable **HOME**, de la que ya hemos hablado y que se refiere al /home/usuario de cada uno de los que tengan cuenta en el sistema. *figura 5*.

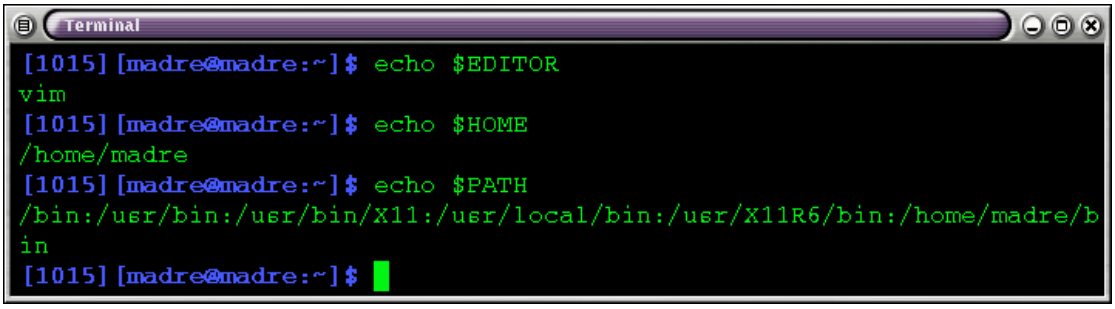


figura 5

Observe el uso del carácter **$**; Indica el principio del nombre de una variable *shell*, sin él, la orden **echo** de volvería únicamente la cadena de texto pasada a continuación. De manera que siempre que queramos ver el contenido de una variable (**o llamarla desde la línea de comandos o desde algún script)**, debemos usar **$**.

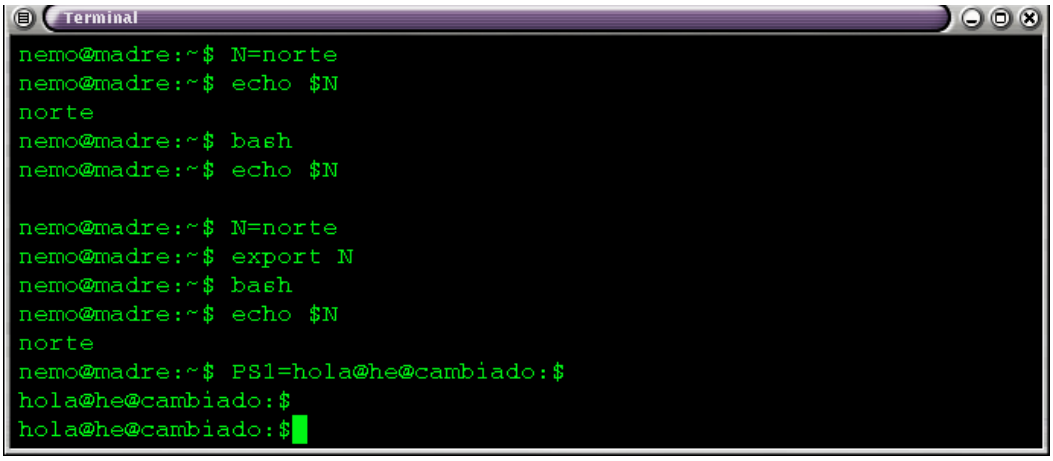


figura 6

DEFINIR VARIABLES

En la *figura 6* definimos una variable que tiene un nombre: **N** (por convención el nombre de las variables se escribe en mayúsculas), y un valor: **norte**. Nótese que no hay que dejar espacios entre el nombre, el igual y el valor. Ahora podemos usar el valor de esta variable refiriéndonos a ella por su nombre.

Lo importante aquí es: ¿donde?, pues en la ejecución de ordenes desde el prompt, por ejemplo, o en el uso de scripts de shell. Pero, ojo, las variables de usuario creadas de esta manera solo funcionan en el *shell* (o *subshell*) en que se han definido. Como vemos en el ejemplo de la *figura 6* si comprobamos el valor de **N** vemos que está correcto, pero después hemos lanzado otro *shell* que se superpone a la anterior *(con el comando **bash**)* y aquí no aparece. La variable a solo es "visible" desde la *shell* donde se fue definida; Es una **variable local**.

Si queremos ver las variables que tenemos en cada shell podemos hacerlo con la orden **env** veremos una lista de variables iniciales del entorno o ***globales*** y con la orden **set** veremos una lista de variables ***locales***. Resumiendo, en cada shell se tendrá acceso únicamente a las variables locales que tenga definidas. Esto es importante de cara a la ejecución de scripts que, por algún motivo, requieran el uso de una variable determinada, o al revés: ¿que pasa si queremos que nuestra variable sea accesible por programas ejecutados en otros shells?

EXPORTAR EL ENTORNO

La solución a este pequeño problema es lo que se conoce como exportar variables. El comando **export** es capaz de hacer dos cosas:

1.Sin argumentos muestra una lista con las variables exportadas.

2.Convierte las variables consignadas (locales) en globales, para que sus valores sean accesibles por cada *shell* que se genere. En el ejemplo de la *figura 22* usamos el comando export para exportar la variable **N**.

Ahora el valor de la variable a puede ser leído desde otro *shell* o *subshell* y también por programas que se ejecuten en ellos.

CREACIÓN DE ALIAS

Los alias son definiciones de comandos en un formato más corto y son definidos por root en el archivo ***/etc/bashrc*** (cuando deben afectar a todo el sistema)**,** y en los ficheros ***/home/usuario/.bashrc*** si sólo son para uso de un usuario en particular**.** Se pueden ver algunos alias muy comunes si abrimos el fichero **/home/usuario/*.bashrc***. (En algunas distribuciones, como en las Debian se definen en el **/home/usuario/*.bash_profile***). Cada usuario tiene sus preferencias y sus costumbres, personas que llegan a GNU/Linux desde otros sistemas operativos y llevan años tecleando: **l** para listar directorios pueden seguir haciéndolo definiendo un alias como: **alias l=ls -lai** .

SISTEMA X-WINDOW

Al igual que en otros sistemas operativos, en los últimos años han proliferado sistemas de ventanas facilitar al usuario la utilización del sistema, además de ser una buena forma de vender el producto.

Desde el sistema operativo MAC que con su MAC/OS inicio esta guerra, Microsoft con su Windows, Linux y UNIX crearon el sistemas de ventanas X-Windows.

Si queremos podemos utilizar un equipo como servidor X-Windows, así los demás equipos podrán conectarse a este servidor y poder ejecutar la aplicación gráfica.

INTRODUCCIÓN A LOS REQUERIMIENTOS DE HARDWARE

Para que el sistema de ventanas funcione tenemos que tener instalado en nuestro equipo una tarjeta de vídeo que sea compatible con las especificaciones del sistema.

Linux soporta todas las tarjetas de vídeo estándar Hércules, CGA, EGA, VGA, IBM monocromo y Súper VGA. Se soportan prácticamente todas la tarjetas actuales, varias aceleradoras 3D (las Vodoo, etc.?) y tarjetas AGP.

En las versiones 3.3.x de X Window, cada chip gráfico corresponde con un servidor X , así a las tarjetas SVGA le corresponde el XF86_SVGA, a las que son ATI Graphics les corresponde XF86_Mach64.

En caso de utilizar una tarjeta que no este dentro de la lista, casi siempre podremos asociarla con una tarjeta estándar aunque perdamos por ello calidad.

Si tampoco funciona así, puede utilizar el servidor de **framebuffer** el cual soporta todas las tarjetas compatibles VESA o las que tengan un driver programado, pero perderá las funcionalidades de los server específicos

La lista completa de todas las tarjetas gráficas soportadas esta en http://www.xfree86.org/cardlist.html.

Las versiones 4.0.x (de las que hablaremos), no es necesario instalar un servidor gráfico para cada chip (para cada marca de tarjeta). El servidor siempre es el mismo y habrá que especificarle el driver apropiado al chip de la tarjeta en el fichero de configuración, que en estas versiones se llama: ***/etc/X11/XF86Config-4***, en lugar de: ***/etc/X11/XF86Config***, como en las versiones 3.3x.

En algunas ocasiones, será necesario compilar el kernel habilitando el soporte para una determinada tarjeta gráfica. Los núcleos que compilados que se sirven con las distribuciones, suelen incluir soporte para las tarjetas de uso generalizado. Algunos de estos núcleos no tienen compilado el soporte para el servidor de **framebuffer**, si lo necesitamos, habrá que incluirlo.

INSTALACIÓN

Prácticamente todas las versiones actuales de cualquier distribución GNU/Linux, instalarán por defecto el entorno gráfico. Es decir, las librerías básicas para el sistema gráfico **X Window**, el servidor de ventanas **xfree86** (conocido como servidor **X**, en sus versiones 3.3.x o 4.1.x dependiendo de lo actual que sea la distribución que se esté instalando), así como uno o varios "entornos de escritorio" con sus "manejadores de ventanas", además de otros manejadores de ventanas.

En el caso de que el **X Window** no se hubiese instalado en el proceso de instalación general del sistema operativo. O si por algún motivo se haya desinstalado o se haya corrompido, o si simplemente queremos actualizarlo, deberemos acudir al conocido **apt**. (En el caso de que la distribución utilice paquetes.deb, es decir, sea Debian o esté basada en Debian, si la distribución está basada en RPM, será necesario buscar los paquetes adecuados, las dependencias de éstos y los paquetes necesarios para reemplazar a aquellos que causen algún conflicto con los que vamos a instalar). En primer lugar localizaremos los paquetes con la orden: **apt-cache search xfree** y **X window**. Además de una gran cantidad de librerías y programas, encontraremos dos que son el corazón del sistema gráfico. Procederemos a su instalación:

apt-get install x-window-system

apt-get install xserver-xfree86 (versión 3.3.x o 4.1.x)

Una vez terminada la instalación de estos dos paquetes (junto a muchos otros de los que dependen, **apt** lo hará automáticamente), disponemos del servidor X, ahora debemos instalar un escritorio o algún manejador de ventanas, porque en realidad el servidor X sólo sirve las ventanas, pero no sabe que hacer con ellas. Detengámonos un instante en este punto.

Una aplicación, por ejemplo Mozilla un navegador web (gráfico), muy utilizado en Linux (aunque en Windows también va muy rápido), necesita mostrarse a si mismo. Así, cuando el usuario lo lanza, Mozilla solicita a xfree que mande las ventanas (la información que contiene los datos de su aspecto), a la salida standard que en este caso siempre será la pantalla del monitor (más adelante veremos que puede ser a un Display determinado). En todo caso, las ventanas del navegador serán procesadas por la tarjeta gráfica y el servidor X las enviará al lugar que en el **XF87Config-4** definimos como **Screen** (con las resoluciones que allí definimos) y al monitor que también definimos en el mismo archivo de configuración.

Pero, el servidor X no sabe que aspecto tienen las ventanas del Mozilla, en realidad el sólo se encarga de transportarlas con la mayor eficacia posible. Del aspecto que deben tener estas ventanas se encargan los manejadores de ventanas (gestores de ventanas). Estos son los encargados de dibujar las ventanas en sus formas, colores y decoraciones.

Recogen la información que transporta el servidor X y la dibujan para que el hardware (en este caso el monitor), pueda mostrarlas.

Es necesario pues instalar alguno (o varios) de estos gestores de ventanas, de ellos hablaremos una vez hayamos visto la configuración del servidor gráfico.

HERRAMIENTAS DE CONFIGURACIÓN

Antes de ver las diferentes aplicaciones que podemos utilizar para configurar nuestra máquina vamos a ver cuales son los datos que hay que conocer.

- Especificar el ratón que queremos utilizar.

- Opciones del teclado que queremos utilizar.

- Opciones de refresco y características del monitor.

- Opciones de la tarjeta gráfica con su chipset y memoria.

- Configuración de las opciones de resolución de las imagenes.

Existen varias herramientas para configurar el servidor de X-Windows, aunque lo mejor es editar directamente el fichero ***/etc/X11/XF86Config-4***

1. **Xf86config:** Es un programa en modo texto. En diferentes apartados tendremos que ir configurando el ratón, el teclado, la tarjeta gráfica con su memoria, el chipset que utiliza, la resolución gráfica que utilizaremos, etc.

2. **Xconfigurator:** También en modo texto pero mas amigable para configurar, este programa intenta detectar que tarjeta tenemos instalada y la configuración optima para nuestro servidor X. En el caso de fallar, podemos hacerlo manualmente mediante la selección de la tarjeta a través de una lista de tarjetas soportadas. (Solo para distribuciones basadas en Red Hat).

3. **xf86cfg:** Esta utilidad en modo gráfico, ofrece la oportunidad de configurar el teclado, el ratón, el monitor y la tarjeta gráfica. Permite mover el puntero a través del teclado y editar el fichero: ***/etc/X11/XF86Config-4***. Pero, quizá la labor más importante que

realiza es que después de scanear el hardware de la máquina, crea un fichero base (ejemplo), del ***/etc/X11/XF86Config-4***, al que llama ***XF86Config.new***. Deja este fichero en el directorio desde el que se ejecutó la orden: **xf86cfg** y en la mayoría de las ocasiones, basta con renombrarlo:

mv XF86Config.new XF86Config-4

sustituyendo al ***XF86Config-4*** original (no estaría de más hacer una copia de este último antes de machacarlo), para que el servidor X funcione correctamente. En otras ocasiones es necesario hacer alguna pequeña modificación en el ***XF86Config.new*** para hacerlo operativo. En todo caso comparando el original ***XF86Config-4*** y el nuevo ***XF86Config.new***, podremos construir uno adecuado a nuestra máquina.

FICHERO DE CONFIGURACIÓN

El fichero ***/etc./X11/XF86Config-4*** contiene la configuración que los programas anteriormente mencionados han generado. Este archivo esta dividido en secciones. Estas secciones están señaladas en el código:

Section "Files"
.....
EndSection

```
# File generated by Sergio Rua.
Section "Module"
  Load "glx.so"
   Load        "dbe"   # Double buffer extension

   SubSection  "extmod"
     Option    "omit xfree86-dga"   # don't initialise the DGA extension
   EndSubSection

   Load        "type1"
   Load        "freetype"

# This loads the GLX module
#    Load       "glx"

EndSection
```

Las secciones que tiene este archivo son las siguientes:

Files: Contiene los archivos de configuración del sistema gráfico.

```
#
**********************************************************************
***
# Files section.  This allows default font and rgb paths to be set
#
**********************************************************************
***

Section "Files"
   RgbPath     "/usr/X11R6/lib/X11/rgb"

   FontPath   "/usr/X11R6/lib/X11/fonts/misc/"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
   FontPath   "/usr/X11R6/lib/X11/fonts/Type1/"
   FontPath   "/usr/X11R6/lib/X11/fonts/TrueType/"
   FontPath   "/usr/X11R6/lib/X11/fonts/encodings/"
   FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/"
   FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection
```

ServerFlags: Señala las señales del sistema que se podrán activar o desactivar.

```
#
**********************************************************************
***
# Server flags section.
#
**********************************************************************
***

Section "ServerFlags"

EndSection
```

Input devices: Opciones de configuración del teclado. Se eligirá la velocidad de repetición de teclas, el país del teclado, etc. Así como las características del ratón que utilicemos, los modos del monitor con su frecuencia dependiendo del modo de configuración que escojamos y

también se elegirá la tarjeta gráfica con sus opciones de memoria y el chipset utilizado.

```
#
**********************************************************************
***
# Input devices
#
**********************************************************************
***

Section "InputDevice"
   Identifier  "Keyboard1"
   Driver      "Keyboard"
   Option "AutoRepeat" "500 30"

   Option "XkbRules"   "xfree86"
   Option "XkbModel"   "pc102"
   Option "XkbLayout"  "es"
EndSection



#
**********************************************************************
***
# Core Pointer's InputDevice section
#
**********************************************************************
***

Section "InputDevice"
      Identifier  "Mouse0"
      Driver      "mouse"
      Option      "Protocol" "ImPS/2"
      Option      "Device" "/dev/psaux"
      Option      "ZAxisMapping" "4 5"
#       Option      "Emulate3Buttons" "No"
EndSection



# Any number of monitor sections may be present

Section "Monitor"
      Identifier   "Monitor0"
      VendorName   "HWP"
```

```
        ModelName     "HP D8901"
        HorizSync     30.0 - 70.0
        VertRefresh   50.0 - 120.0
EndSection

#
**********************************************************************
***
# Graphics device section
#
**********************************************************************
***

# Any number of graphics device sections may be present

# Standard VGA Device:

Section "Device"
    Identifier  "FrameBuffer"
    VendorName  "Unknown"
    BoardName   "Unknown"
    Driver      "fbdev"
EndSection

# Device configured by xf86config:


Section "Device"
        Identifier  "Intel"
        Driver      "i810"
EndSection

Section "Device"
        Identifier "VGA"
        Driver     "vga"
EndSection

Section "Device"
        Identifier  "Card0"
        Driver      "mga"
        VendorName  "Matrox"
        BoardName   "MGA G100 AGP"
        BusID       "PCI:1:0:0"
                EndSection
```

Screen: En este apartado elegiremos los diferentes servidores que se utilizarán para los diferentes modos de configuración en cuanto al tamaño de la imagen utilizado. (Nótese que aunque en el apartado anterior hemos nombrado varios dispositivos gráficos, es en este apartado donde elejimos uno de ellos: Card0, que corresponde a una tarjeta Matrox). .

```
#
**********************************************************************
***
# Screen sections
#
**********************************************************************
***

Section "Screen"
   Identifier  "Screen 1"
   Device      "Card0"
   Monitor     "Monitor0"
   DefaultColorDepth 16
   Subsection "Display"
      Depth   16
      #Modes       "1600x1200" "1280x1024" "1280x960" "1024x768" "800x600"
      Modes       "1152x864" "1024x768" "800x600"
      ViewPort    0 0
   EndSubsection
EndSection
```

ServerLayout: Aquí elegimos el Screen, el ratón y el teclado entre los editados anteriormente.

```
#
**********************************************************************
***
# ServerLayout sections.
#
**********************************************************************
***

Section "ServerLayout"
   Identifier  "Simple Layout"
   Screen "Screen 1"
   InputDevice "Mouse0" "CorePointer"
   InputDevice "Keyboard1" "CoreKeyboard"
EndSection
```

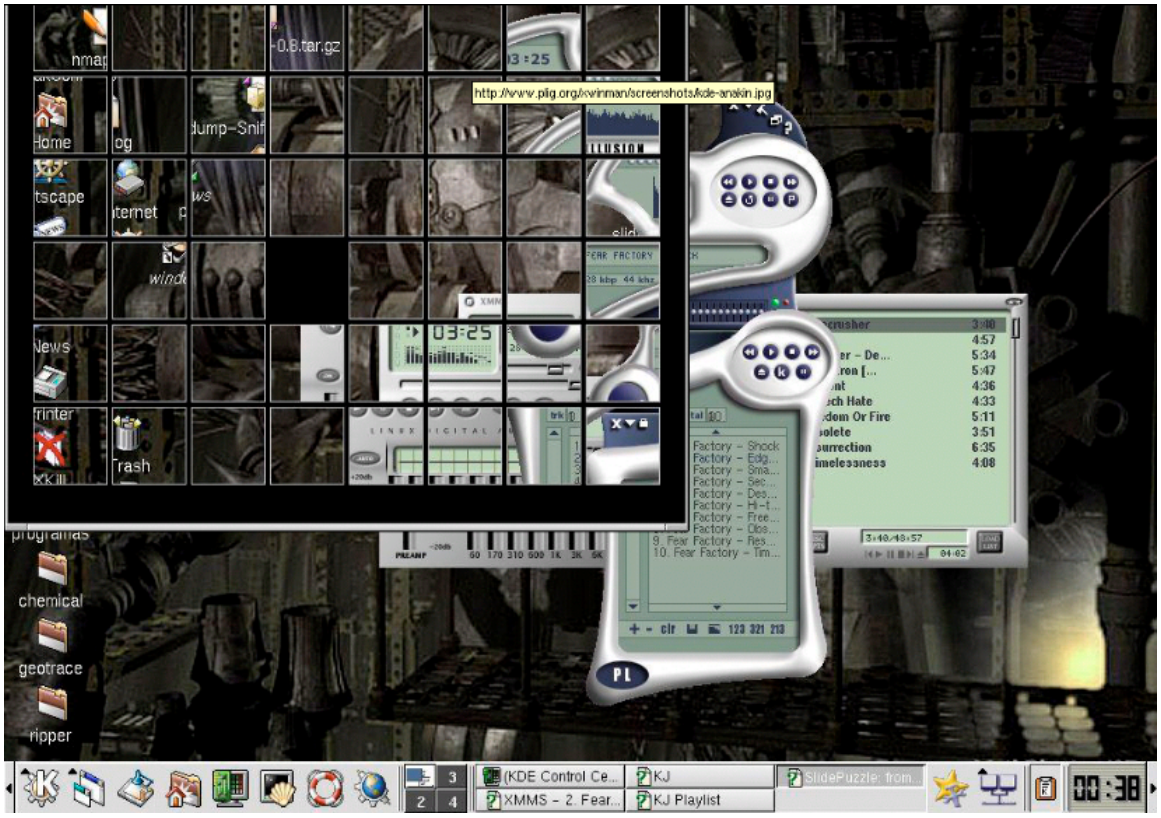
ESCRITORIOS Y MANEJADORES DE VENTANAS

Durante estos últimos años se ha avanzado mucho en el desarrollo de un entorno gráfico para GNU/Linux. En realidad se desarrolla tanto que en este momento, existen al menos dos completos escritorios y casi una docena de gestores de ventanas (por hablar sólo del software Open Source).

Los escritorios **KDE**, **Gnome, XFce** y **CDE** (este último, utilizado por los **UNIX** propietarios de **IBM**, **HP**, **Compaq**..), proveen al usuario y cada día más al administrador, de todas las aplicaciones que pueda necesitar y algunas que probablemente, la mayoría, no necesite nunca. Y todas ellas integradas en un entorno gráfico muy intuitivo, amigable y por lo tanto fácil de usar. Algo que los usuarios finales de **Linux** agradecen mucho.

Por lo general uno de estos escritorios, se instalan por defecto con cualquier distribución de **GNU/Linux**. Si necesitamos instalarlas en un sistema en producción, debemos acudir de nuevo a **apt**, buscar los paquetes necesarios (casi siempre metapaquetes) y una vez localizados, instalarlos con la orden: **apt-get install**.

Estos escritorios son diferentes en cuanto a su aspecto (también en su código de programación, utilizan librerías muy diferentes. Algunas de las librerías utilizadas por los desarrolladores de **KDE** no eran del todo Open Source, hasta hace pocos meses). **KDE** admite gran variedad de "temas" que, básicamente, cambian colores y decoraciones de las ventanas. Su aspecto recuerda un poco al escritorio de Windows y quizá por ello es uno de los preferidos por los usuarios recién llegados a **Linux**. Algunas distribuciones lo instalan y lanzan por defecto.

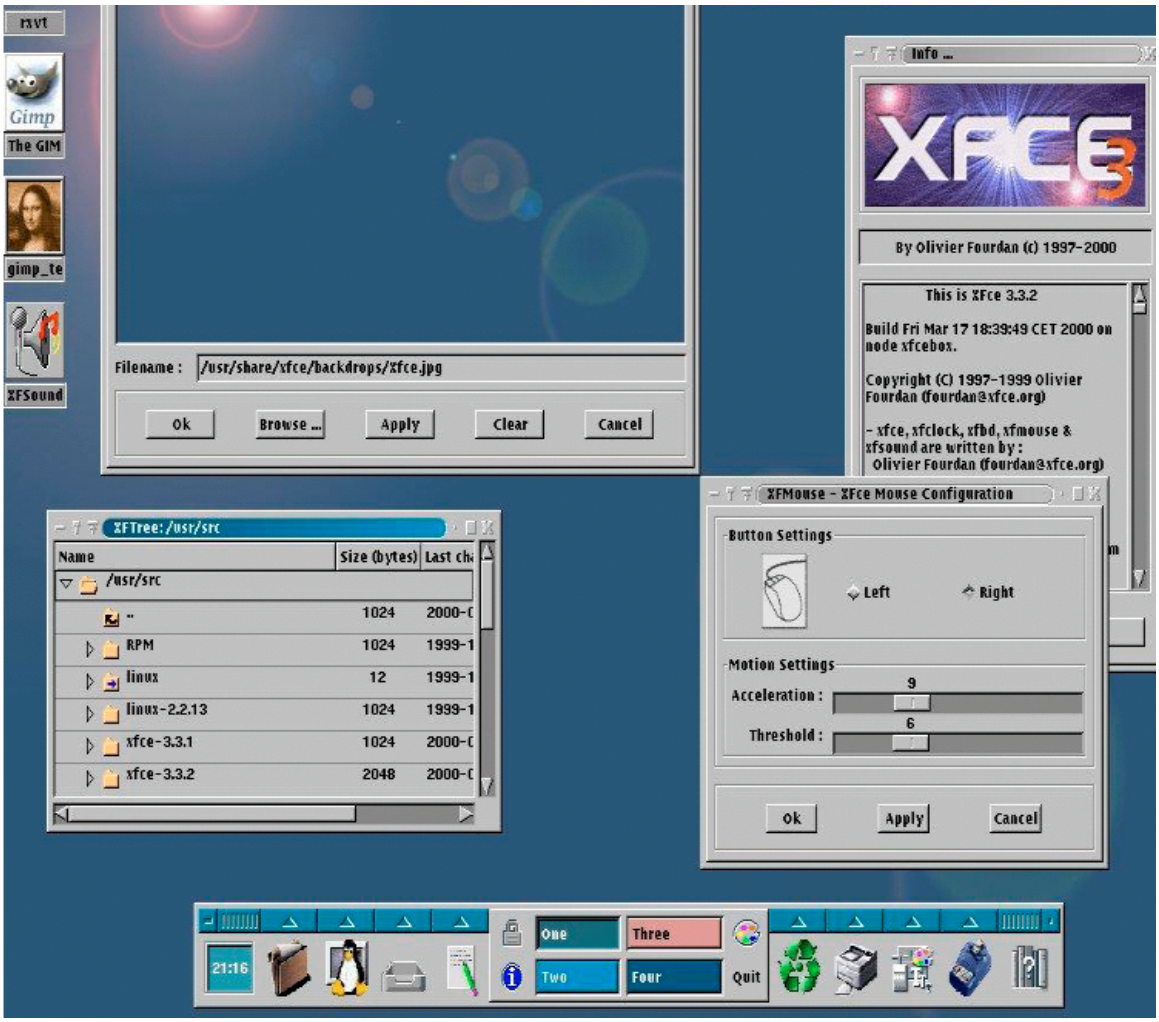

http://www.plig.org/kwinman/screenshots/kde-anakin.jpg
Trash
chemical
geotrace
ripper
(KDE Control Ce...
KJ
XMMS - 2. Fear...
KJ Playlist
SlidePuzzle: from...
03:38

KDE2

Gnome es mucho más configurable, admite el uso de varios manejadores de ventanas "externos", como **Window Maker**, **Enlightenment**, **Sawfish**, **IceWM**, **AfterStep**, **Blackbox**, **FVWM** y otros. Algunos de estos gestores no sólo cambian el color y las decoraciones de las ventanas, sino también su comportamiento. **Gnome** fue durante algunos años el escritorio favorito de los usuarios de **GNU/Linux**, hoy prefieren otros entornos de escritorio y manejadores de ventanas que consumen menos recursos (menos consumo de RAM y menos tiempo de CPU).

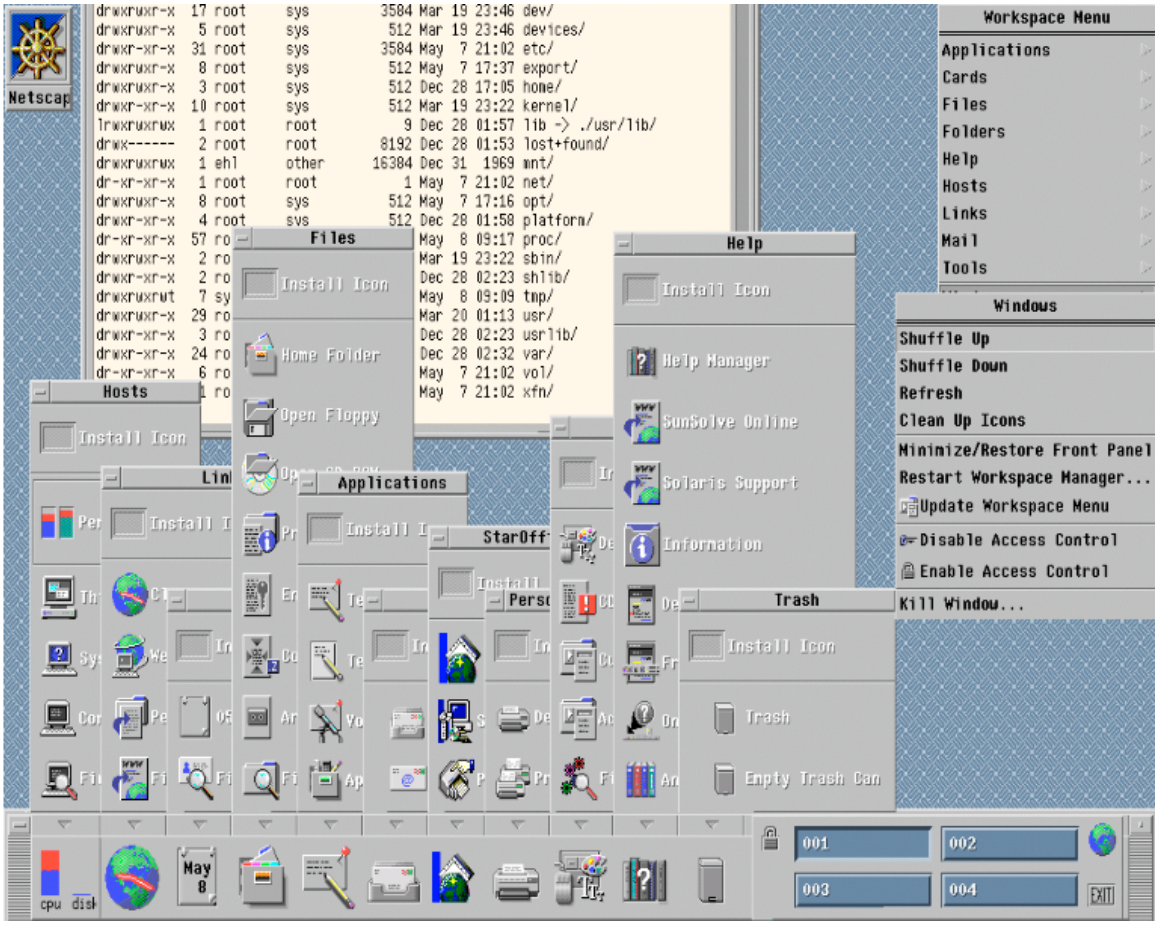
Gnome

XFce es un entorno de escritorio muy ligero para **UNIX**, muy parecido al originario **CDE** comercial. Ahora está construido con las librerías **GTK+** y en los últimos tiempos está siendo muy utilizado por mochos usuarios de **GNU/Linux**, sobre todo por aquellos que administran máquinas con recursos limitados. Es muy fácil de configurar y dispone de un manejador de ventanas, **Xfwm** y de algunas buenas herramientas como el navegador de archivos **XFTree** y un buen navegador para Samba, **XFSamba**. También dispone de un módulo de la conformidad de **GNOME**.



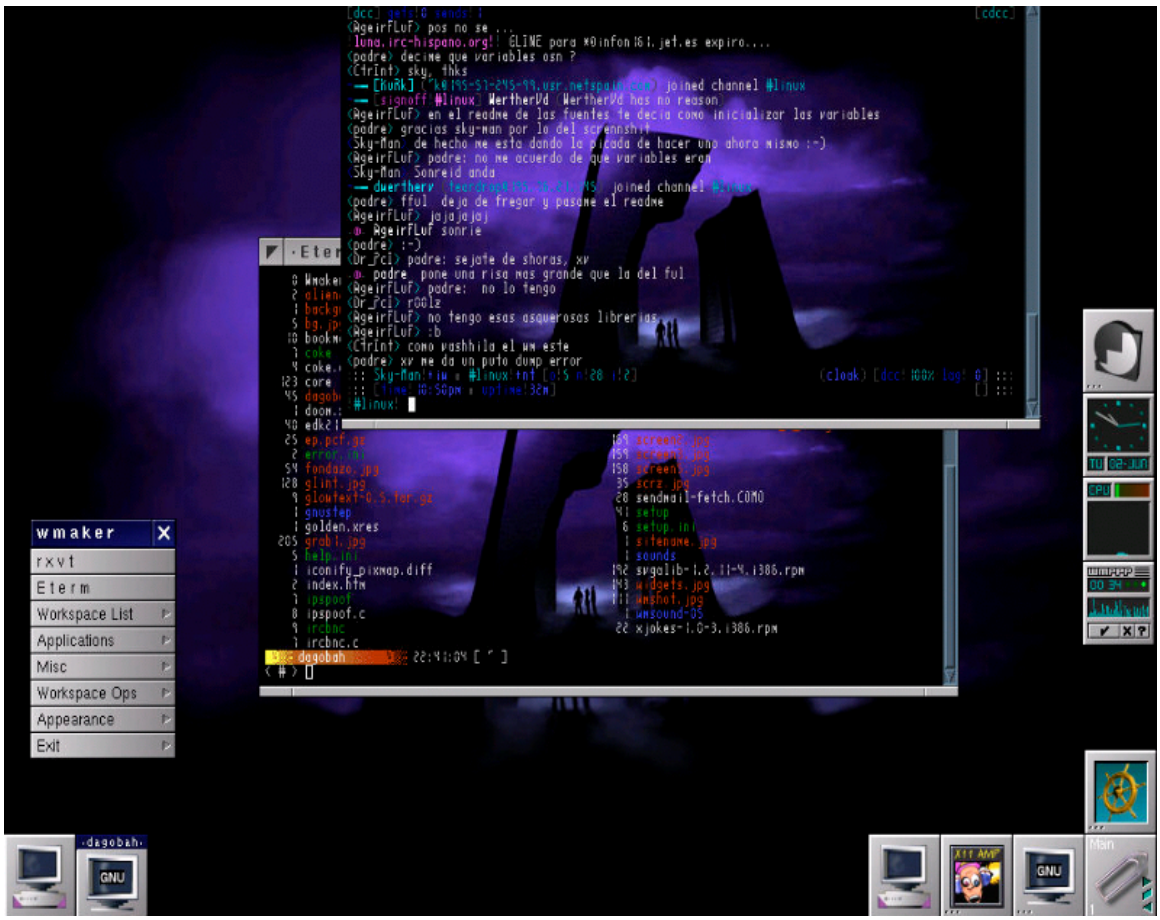
XFce

CDE o el entorno de escritorio común, es un interfaz gráfico comercial para **UNIX**. Lo utilizan (**AIX**, **Digital UNIX**, **HP/UX**, **Solaris**, **UnixWare** de **SCO**). Este escritorio ha sido desarrollado en común por Hewlett-Packard, IBM, Novell y el sun Microsystems. Ha sido adoptado como un entorno más o menos estándar por estas compañías y otras en el **UNIX**. Obsérvese su parecido con **XFce**, en esta captura de **Sun Solaris**.



CDE

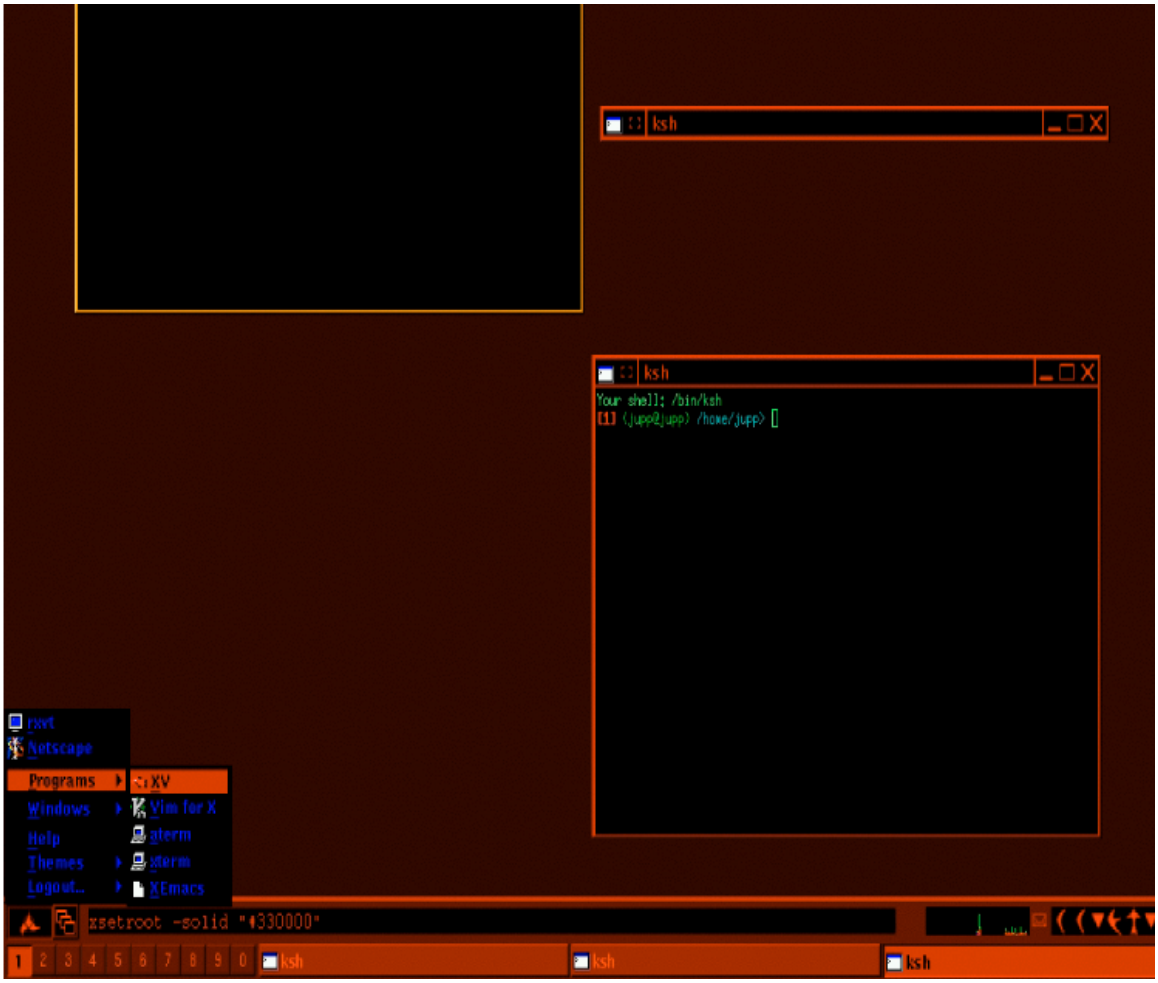
Como ya se ha dicho, existen otros programas a los que se les llama habitualmente gestores de ventanas (window managers), que pueden ser utilizados por si mismos, es decir, sin integrarlos en ninguno de los escritorios mencionados anteriormente. Son muchos, por destacar algunos de los que siguen siendo desarrollados (actualizados), nombraríamos: **Window Maker**, **IceWM**, **AfterStep**, **Blackbox** y **FVWM**. Se puede decir que todos estos manejadores de ventanas "independientes" de entornos de escritorio, consumen pocos recursos (aunque algunos de ellos necesitan cargar librerías de **Gnome**). También es cierto que la mayoría de ellos no son demasiado intuitivos para aquellos usuarios procedentes de otros entornos (Mac o Windows). Esta desventaja se puede aplicar también a su configuración.



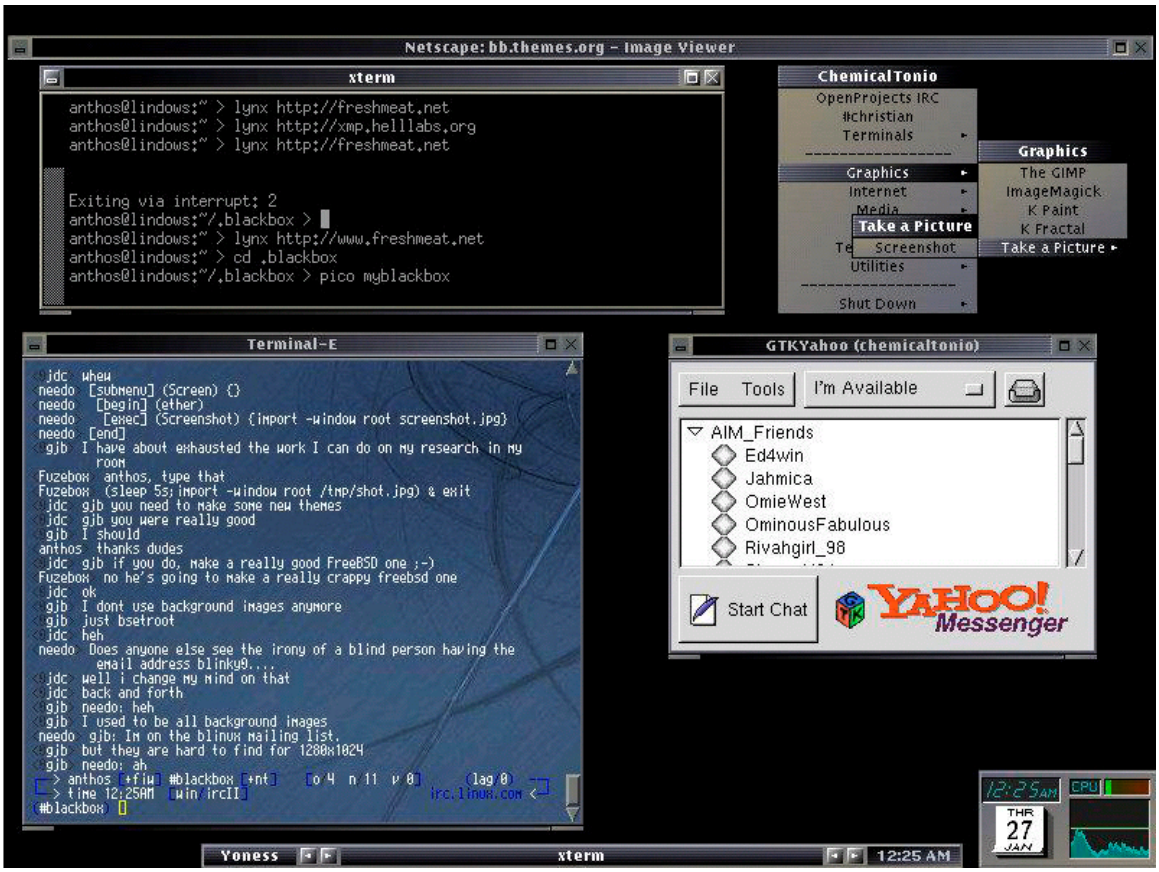
Window Maker


rxvt-xpm
trol Panel
: Untitled
.steprc
NCFtp
Please use a mirror if possible
FTP Site
Directory
UK
sunsite.doc.ic.ac.uk /packages/linux/redhat
nfs-uxsup.csx.cam.ac.uk /pub/redhat/
ftp.ox.ac.uk /pub/linux/redhat
Europe
ftp.ktnet.co.kr /pub/redhat
linux.stt.it /pub/mirrors/redhat
ftp.osb.hu /pub/mirrors/linux/redhat
sunsite.icm.edu.pl /pub/Linux/redhat
ftp.vma.bme.hu /pub/linux/redhat
ftp.sfos.ro /mirror/RedHat
ftp.u-bordeaux.fr /pub/Linux/redhat
rxvt-xpm
leo [ximenes ~]$
File Modules He
File System Configuration
Package Management
Search Help Systems
Kernel Daemon Configuration
Writer 3.1
xv

AfterStep


ksh
ksh
Your shell: /bin/ksh
[1] (jupp@jupp) /home/jupp
rxvt
Netscape
Programs
Windows
Help
Themes
Logout...
XV
Vim for X
aterm
xterm
XEmacs
xsetroot -solid "#330000"
1 2 3 4 5 6 7 8 9 0
ksh
ksh
ksh

IceWM



BlackBox

CREACIÓN DE SHELL SCRIPTS (O ARCHIVOS DE COMANDOS)

La shell más utilizada en Linux es un programa compilado (**Bash**). La mayoría de los usuarios de **GNU/Linux** y todos los administradores, lo utilizan habitualmente. Es el intérprete de comandos, proporciona un interface interactivo que permite ejecutar comandos y ver los resultados.

Pero, como ya dijimos anteriormente, **Bash** proporciona un potente lenguaje interpretado que permite:

-Manipular la cadena de carateres
-Construcciones de control de flujo (fi, while, for ..)
-Comunicación entre procesos a través de señales, argumentos y salida (exit)
-Cualquier modificación de ficheros
-Generar listas de argumentos
-Definición de funciones

Algunas de estas propiedades ya han sido explicadas a lo largo de este curso, otras más avanzadas son las que analizaremos a partir de ahora.

En realidad podemos hacer uso de la capacidad de programación de **Bash**, desde la línea de comandos, pero su potencia está en la capacidad de leer los comandos desde ficheros (scripts de shell). Esta habilidad permite guardar los comandos en ficheros para utilizarlos con posterioridad.

EJECUCIÓN DE SHELL SCRIPTS

Más adelante se crearemos scripts utilizando cualquier editor (preferentemente **vim**). Una vez creados se pueden ejecutar de tres formas distintas:

sh nombre del script

Para ejecutarlo mediante esta orden, el fichero debe de tener permisos de lectura. Al lanzarlo de esta forma, **se genera una shell hija desde la que se ejecutan los comandos**. Es una de las formas más habituales de lanzar los scripts, cuando no están suficientemente depurados.

nombre del script

Para ejecutarlo mediante esta orden, el fichero debe de tener permisos de ejecución. Es la forma habitual de lanzar scripts de shell. **También en este caso se**

genera una shell hija que hereda el entorno de la shell padre. Si el script que se lanza modifica el entorno de trabajo, éste sólo será modificado para la shell hija que acaba de generar, por lo que tampoco afectará a ningún proceso dependiente de la shell padre. De esta forma se garantiza una total independencia entre los procesos que se ejecutan desde una misma shell.

.nombre del script

Para ejecutarlo mediante esta orden, el fichero debe de tener permisos de lectura. Lanzando el script de esta forma (no olvidar el punto anterior al nombre del script), no se genera la shell hija.

Los scripts deben de lanzarse de esta forma cuando se va a modificar el entorno de la shell. Así todas las modificaciones que el script realice en el entorno se transmitirán a todos los procesos hijos de la shell. Algunas de estas modificaciones pueden ser tan importantes como un cambio en la variable **PATH**.

Un ejemplo de utilización de esta forma de ejecución, es el que hace el sistema al ejecutar los scripts contenidos en los ficheros: **/etc/profile** y **$HOME./bash_profile**. En estos ficheros hay guiones de shell que modifican el entorno global (de todos los usuarios), o el entorno particular (de un usuario).

exec nombre del script

Para ejecutarlo mediante esta orden, el fichero debe de tener permisos de ejecución. Lanzando el script (o cualquier otro programa) de esta forma, el shell padre se sustituye por el programa lanzado.

Un ejemplo de este comportamiento podemos encontrarlo también en el propio sistema. Cuando se escribe el nombre de un usuario válido en la línea de comandos (o en un programa de login gráfico), se está ejecutando un proceso **getty** (login), después de introducir correctamente la contraseña, el proceso **getty** es sustituido por el proceso **bash**.

El sistema utiliza esta técnica internamente para lanzar muchos procesos. Si desde la shell lanzamos un proceso de esta forma, ésta será sustituida por el proceso lanzado, así, cuando el programa termine terminará también la sesión. Al terminar la sesión (normalmente), se ejecutaría de nuevo el proceso **getty** (login), porque así se establece en el fichero **/etc/inittab** que, en cierta forma, también es un script de shell.

Se suele utilizar esta técnica, para lanzar aplicaciones desde el

$HOME/.bash_profile de un usuario. De esta forma cuando el usuario termine de trabajar con su aplicación, la aplicación se cierre por cualquier motivo o el usuario se salga de ella, regrese siempre al login. Si ni se hace así, al acabar la aplicación el usuario saldría a la shell padre, algo peligroso si el programa se ejecutaba con permisos de administración. En muchos casos podemos desear que un usuario o un grupo de ellos, sólo puedan ejecutar unas determinadas aplicaciones y ninguna otra, utilizando esta técnica impedimos que ocurran desastres irreparables.

PARA QUÉ SE UTILIZAN LOS SHELL SCRIPTS

Una de las razones más comunes para la utilización de scripts es la comodidad. Algunos comandos requieren tantas opciones que las órdenes se hacen muy largas. Si los usamos con frecuencia, es mejor escribirlos en un fichero y ejecutar después ese fichero.

Como ejemplo podemos utilizar el comando **find**. Es un caso claro de orden muy utilizada, cuya sintaxis más simple es, demasiado larga:

find / -name se_busca -print 2>/dev/null

Find es el comando que utilizamos para buscar archivos, **/** con este símbolo le indicamos que busque en el sistema de archivos raíz. La siguiente opción **-name** indica que debe buscar lo que escribamos a continuación, que debe de ser el nombre del archivo que queremos localizar. La opción **-print** indica que nos muestre el resultado en pantalla. Por último redirigimos la salida standard de errores (2>staderr), a **/dev/null**, nuestro agujero negro particular, para que no nos muestre por pantalla los mensajes de error . Si un usuario busca un archivo en el sistema de ficheros raíz, el comando **find**, tendrá que entrar en muchos directorios y ficheros que no tienen permiso de lectura para ése usuario, por lo que el sistema indicará un: **permiso denegado**. Podemos librarnos de estos mensajes enviándolos a **/dev/null**, como en la orden anterior o, mandarlos a un archivo que más tarde leeremos (porque a lo mejor nos interesa precisamente eso, los mensajes de error).

Además de lo largo que resulta la orden anterior, tenemos algunos inconvenientes añadidos. Por ejemplo, que cada vez que lo utilizamos (probablemente), buscamos un fichero diferente, por lo tanto donde dice: **se_busca**, debería de figurar la cadena de caracteres que buscamos. También es posible que no se quiera hacer la búsqueda en el raíz (**/**), por lo que habría que indicar otro directorio o sistema de ficheros.

Otro motivo por el que podemos necesitar hacer un script (lo veremos en los ejemplos), es la ejecución de tareas repetitivas en las que se usan un buen número de comandos.

También podemos hacer scripts como forma de crear comandos propios, con nombres más familiares. Aunque para este tipo de actuaciones, podemos utilizar los **alias** en los ficheros de perfil de los usuarios que lo necesiten o incluso con enlaces dinámicos a los comandos que se quieran "renombrar".

Un motivo importante por el que podemos hacer un guión de comandos, es el de sustituir al propio intérprete de comandos (shell), por un interface de menús. De esta forma, facilitamos el trabajo de los usuarios, si además lanzamos estos scripts desde los ficheros de perfil de estos usuarios precedidos del parámetro **exec**, nos aseguramos de que sólo ejecutarán lo que pongamos en esos menús o regresarán al login.

Se suelen usar scripts de shell para aquellos trabajos que se planifican como tareas en segundo plano. La simplificación de trabajos administrativos habituales y por lo tanto repetitivos.

Desde hace ya bastantes años se viene oyendo que la programación shell tiende a desaparecer, o a ser poco utilizada. Lo cierto es que se sigue utilizando mucho. El motivo es en definitiva el que mencionábamos al principio de este capítulo: comodidad.

Los guiones de shell son fáciles de escribir, también son fáciles de leer y se pueden modificar en cualquier momento. Esta misma facilidad propicia que un script de shell se pueda escribir muy rápido, con lo que la respuesta a un "gran problema" puede resolverse a menudo con un "pequeño script".

Por supuesto hoy existen otros lenguajes de programación (incluso para hacer scripts com Python o Perl), a los que el shell no puede sustiuir. Hay que tener en cuenta que la shell, el interprete de comandos que todos usamos, es ya de por si un programa grande. Hasta la generalización de los entornos gráficos, el intérprete de comandos era probablemente el programa que más recursos consumía (en estaciones de usuario ..). Con los recursos de los que se dispone hoy en día (abril de 2002), la shell no es ningún lastre para una máquina moderna. Pero esto no quiere decir que se deba abusar del lenguaje interpretado. Si un script se puede hacer en una línea programando en **Python**, por qué hacerlo en 55 líneas programando en **shell**.

Antes de seguir adelante, explicando conceptos como: procesos padre, proceso hijo, spawn (desovar), procesos síncronos, procesos asíncronos, subshell. Llamadas **fork**, **exec** o **wait**. Entrada y salida standard y los números descriptores de los ficheros de entrada y salida. Construiremos un pequeño script de shell, porque, la práctica es imprescindible.

A cada paso iremos escribiendo otros guiones de shell, que ayudarán a entender los conceptos que se explicarán.

SCRIPT 1

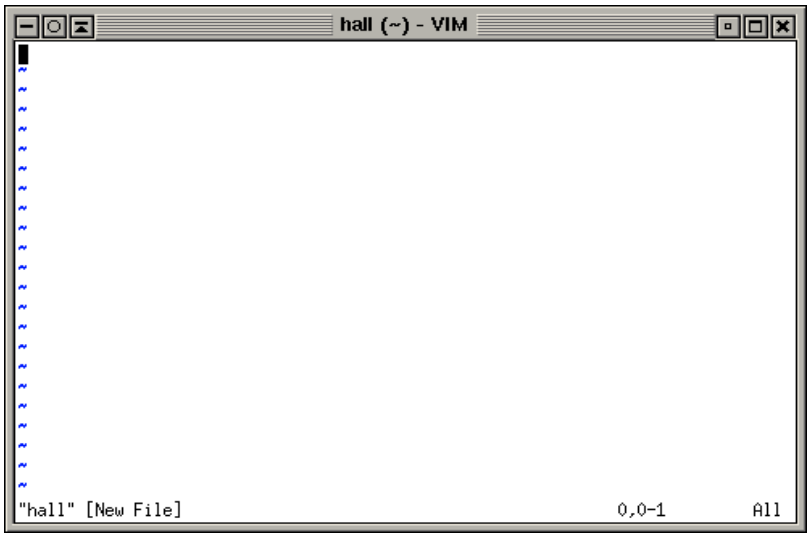
Para empezar cualquier script de shell, es necesario abrir un archivo vacío que llevará como nombre el que hayamos escogido para el script en cuestión. Esto lo haremos lanzando uno de los editores disponibles en cualquier **Linux**, **vim** o **vi** (este último por estar presente en cualquier sistema **UNIX**), son los más aconsejables.

Como práctica, escribiremos un script que ejecute la orden **find** anteriormente nombrada y a continuación comentaremos todas sus líneas y comandos. Es importante escoger el nombre del archivo de comandos. En este caso, el trabajo que realizará el script es buscar, por lo tanto elegiremos un nombre que sea fácil de recordar, que no sea igual a ninguno de los comandos que incluye el guión y por supuesto que sea más corto que la orden a la que se pretende sustituir.

Una vez escogido el nombre (lo llamaremos hall, de hallar), comenzamos a escribir el script, en este caso con la orden:

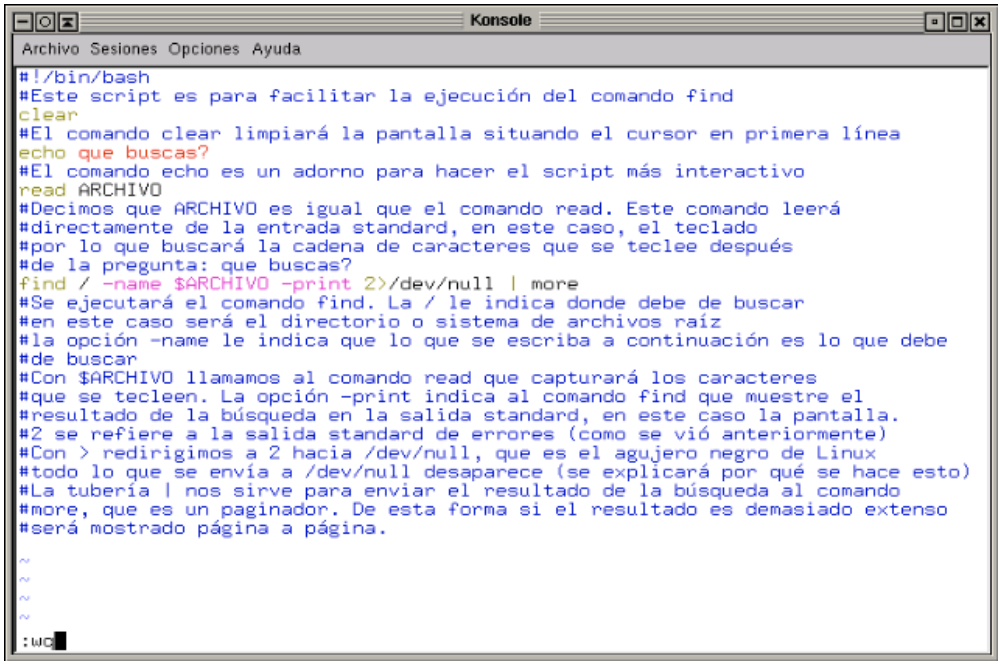
vim hall

Se lanzará el editor **vim**, que mostrará un archivo vacío que tiene como nombre, **hall**.



vim

Escribiremos a continuación todos los comandos que se ejecutarán cada vez que lancemos este script.



hall

Como se puede ver en la figura anterior, el editor **vim**, muestra las líneas en diferentes colores. En realidad se trata de un corrector de sintaxis para código de programación, válido para la mayoría de los lenguajes de programación (consultar: man vim).

En este curso no se tratará extensamente ningún editor, aunque se harán referencias continuas a **vim**. Para conocer más acerca de las capacidades y peculiaridades de este y otros editores, es necesario obtener manuales de uso de los mismos o consultar las páginas del manual en línea de Linux (man).

Si se analizan las líneas de la figura anterior, veremos que la mayoría comienzan con el carácter: **#** . En lenguaje shell, cualquier línea del guión que comience con este símbolo, es un comentario. **Los comentarios son leídos e interpretados por la shell en la que se ejecuta el script. Pero nunca son ejecutados**.

Los comentarios son muy importantes en cuanto a la documentación del script y deben ser los más descriptivos posible. Cualquier usuario que no sepa nada de programación shell, puede sin embargo saber para que sirve el script **hall**, sólo con abrir el script con un editor y leer las líneas comentadas. A veces se olvidan los comentarios o se cree que como el script es tan fácil no son necesarios, cuando en realidad son siempre de gran ayuda.

En la figura que muestra el script **hall**, los comentarios explican claramente que hace cada comando y para que sirve cada opción (el código de colores los destaca). Como nunca está de más, profundizaremos en aquellas líneas que necesitan de mayor explicación.

La primera línea del script, comienza con un comentario: **#!/bin/bash**. Todos los scripts deben de comenzar con este comentario. Indica a la shell que ejecutará este guión, que se trata de un archivo de comandos y que lo que se escribe a continuación es un guión que debe de ir leyendo, interpretando y ejecutando en el orden en el que está escrito. Cuando tecleemos el nombre del script (**hall**), la shell (en este caso bash), buscará hall entre sus comandos internos, al no hallarlo mirará en el **PATH** del usuario que quiere lanzar la orden. Si no lo encuentra o el usuario no tiene permisos para ejecutarlo mostrará un mensaje de error diciendo que no encuentra el comando o que el permiso es denegado (esto es importante y lo veremos más adelante). Si encuentra un archivo binario con permisos de ejecución para ese usuario que se llame hall, leerá esta primera línea y entenderá que es un script y que debe de ir ejecutando los comandos que figuran en el.

En ocasiones a los archivos que contienen scripts de shell, se les añade la extensión **.sh**, esto no es estrictamente necesario. A la shell no le sirve de nada esta extensión y al núcleo de **Linux**, tampoco. Sin embargo otras aplicaciones pueden usar esa extensión para asociar archivos.

Después de escribir el comando **find** con todas sus opciones, redireccionamos la salida standard a /dev/null (**2>/dev/null**). Null es un fichero especial de dispositivo tipo carácter, que en realidad no controla ningún disco o partición. Tiene como misión "fagocitar" todo lo que se escribe en el.

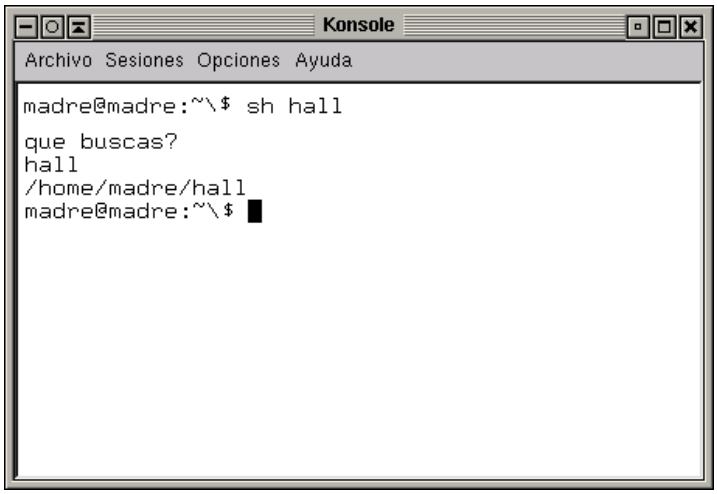
La función que realiza esta redirección en el script **hall**, es la de enviar a /dev/null todos los mensajes de error que mostraría el comando find. Si éste script es ejecutado por un usuario que no sea **root**, aunque el usuario tenga permisos de ejecución sobre el script y sobre el comando find, no tendrá (o no debería) tener permisos de lectura en todos los directorios del sistema de ficheros. Como el comando find buscará en **/** (directorio raíz), es decir, en todos los directorios que contiene y por tanto en todo el sistema, tendrá que "entrar" en esos directorios y al no tener permisos de lectura sobre todos ellos, mostrará un mensaje de error (permiso denegado).

Si no se redireccionan estos mensajes hacia un fichero, serán mostrados en pantalla y acabarán mezclándose con los resultados de la búsqueda. En el caso del script **hall**, redireccionamos estos mensajes a /dev/null, para que no sean mostrados y además no ocupen lugar en el disco. Sin embargo en algunas ocasiones deseamos que el usuario no vea esos errores, pero que no se pierdan para poder consultarlos después. En ese caso redireccionaríamos a un fichero en una ubicación determinada: **2>>/root/finderr**. Al hacer una doble redirección se consigue que los datos enviados al archivo **finderr**, no lo reescriban, sinó que se vayan concatenando (añadiendo).

Una vez completado el script **hall**, lo probremos lanzándolo con la orden:

sh hall

tal y como vimos anteriormente, por si es necesario depurarlo. El resultado debería parecerse al de la figura siguiente. La shell, aunque el fichero **hall** no esté en el **PATH**, genera una shell hija desde la que ejecuta el script y muestra la pregunta: que buscas? (tal y como se escribió en el guión). Introducimos la cadena de caracteres hall y el script se busca a si mismo, se encuentra y muestra el path absoluto con la situación del fichero **hall**.



Usando a hall

Una vez que se comprueba que el script **hall** se ejecuta correctamente, es el momento de darle los permisos apropiados y situarlo en los **paths** de aquellos usuarios que vayan a utilizarlo. Si deseamos que pueda ser usado por la mayoría de usuarios, lo moveremos al directorio **/bin** junto a los comandos de uso común. Si se tratase de un script para uso administrativo (además de modificar los permisos), lo moveríamos al directorio **/sbin**.

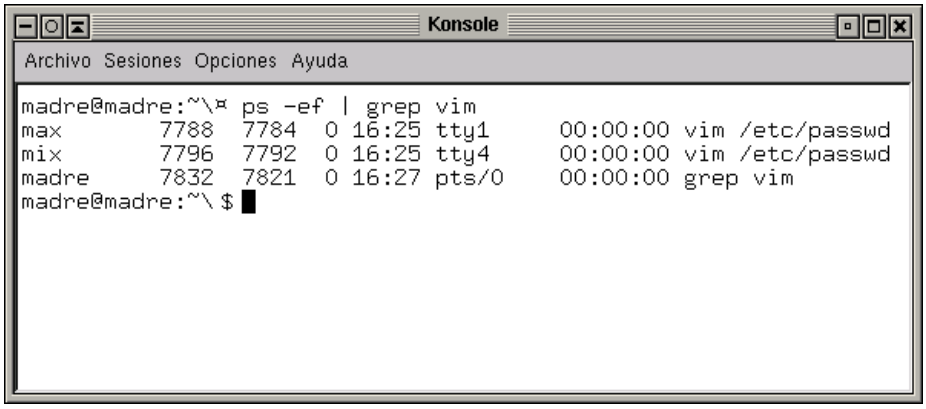
PROCESOS EN LINUX

Un proceso es una instancia de un programa ejecutable escrito para Linux, que incluye todo el código del programa y los datos necesarios para su ejecución. Es frecuente confundir programa y proceso. Vamos a aclararlo.

Como GNU/Linux es un sistema operativo multiusuario y multitarea, varios usuarios pueden trabajar "contra" la misma máquina. Supongamos que el usuario **max** está frente a la máquina **madre** y en un terminal ejecuta el programa (comando.. orden): **cat /etc/passwd**.

Supongamos ahora que el usuario **mix**, se conecta a la máquina **madre** desde otra máquina, a través de **telnet** o **ssh** y ejecuta también la orden: **cat /etc/passwd**.

Si en ese instante lanzamos la orden: **ps -ef |grep vim** (en la figura, procesos), para filtrar los procesos "vim" encontraremos que, al menos, existen dos procesos. Los de los usuarios **max** y **mix.**



Procesos

El sistema operativo **Linux** sólo tiene un editor **vim** instalado y un sólo fichero **passwd**. En realidad el sistema está utilizando el mismo programa **vim** para leer el mismo archivo **passwd**. Lo hace creando dos instancias partiendo del mismo código y poniendo en marcha dos procesos diferentes. Como se puede ver en la figura anterior, los **PID** (número identificador del proceso-segundo campo por la izquierda), son diferentes. Los **PPID** (número identificador del proceso padre), también son diferentes.

Linux crea un nuevo proceso a través de la llamada al sistema **fork**. Cuando se produce esta llamada el sistema hace una réplica del proceso que hace la llamada (es decir, el proceso **padre**), para crear spawn (desovar), un nuevo proceso. (el proceso **hijo**). Dicho de otra forma, si un proceso que está ejecutándose, por lo tanto cargado en memoria quiere llamar a otro proceso (ejecutar otro programa), sin que este último le sustituya en la memoria, hace una llamada **fork.** Al sistema operativo. **En realidad no es otra cosa que hacer una copia de si mismo en la memoria (clonarse) o crear un hijo idéntico al proceso que le creó (a excepción del número de identificación, PID).**

Si la llamada **fork** se ejecuta sin problemas, la misma llamada **fork**, manda al proceso padre el **PID** del proceso hijo y un **0** (cero) al propio proceso hijo. De esta forma los dos procesos sabrán que la llamada **fork** se ha ejecutado bien.

Cuando un proceso "termina", siempre manda un valor numérico a su **PPID**. A este valor se le llama **status de salida**. Si es un **0** indica que el proceso se ha ejecutado sin errores.

El tipo de llamada **exec** superpone el código objeto objeto de un programa, sobre el propio programa que ejecuta la llamada y pasa a ejecutar el código resultante, es decir, el proceso resultante (hijo), sustituye al que ha hecho la llamada (padre). **El proceso padre que se está ejecutando, lanza otro programa y al tiempo muere para que su hijo pueda ocupar el espacio que el tenía en memoria. En algunos casos (como el programa getty), al terminar el proceso hijo que le sustituyo en memoria, resucitan volviendo a crear un nuevo proceso. El caso de getty (proceso login), resucita porque así está escrito en el fichero /etc/initab (que es la tabla de inicio-y de fin- como el libro de la vida para los sistemas GNU/Linux).**

PROCESOS PADRE

Al hacer una llamada **fork**, el proceso padre que la realiza puede hacerlo de varias formas.

Puede usar la llamada **wait** (espera), esto hace que el proceso padre espere a que finalice el proceso hijo. En estos casos el proceso hijo es un proceso **síncrono** con respecto al proceso padre. **Todos los procesos que se lanzan en primer plano (foreground) se ejecutan de esta forma**.

El proceso padre puede continuar ejecutándose sin esperar a que se ejecute el proceso hijo y entonces estaríamos hablando de un proceso hijo **asíncrono** respecto al padre. **Todos los procesos lanzados en segundo plano (background) se ejecutan así.**

Ya hablamos de la forma de lanzar procesos en segundo plano (añadiendo & a continuación de la orden) y de sus ventajas. Una de ellas es la de poder seguir usando la shell en la que se lanza el comando.

LLAMADAS AL SISTEMA

Como hemos visto la shell usa las llamadas al sistema (**fork** y **exec**), para ejecutar otros comandos **GNU/Linux**.

El proceso hijo que se crea cuando la shell usa una llamada **fork** se denomina subshell. Es esta subshell la que se encargará de buscar el comando que se tecleó en la shell padre y ejecutarlo.

Cuando se teclea un comando, los caracteres introducidos en el prompt son también el nombre de un fichero. Si este fichero es un programa compilado (un binario), la subshell usará la llamada **exec** para sustituir su propio código por el del comando y ejecutarlo.

Si el fichero no es un programa compilado, la subshell entiende que es un script. Entonces lo tratará como un guión, leerá y ejecutará las órdenes que estén escritas en ese archivo de comandos, usando un **fork** y un **exec** para cada comando. Es decir, si el script contiene 10 órdenes, la subshell que la shell original creó, irá haciendo un **fork** (creando nuevas subshells) y **exec** desde las nuevas subshells que se crean, para ejecutar los comandos script.

La subshell hará cuantas llamadas fork y exec como sea necesario, creándose de esta forma numerosas subshells descendientes (hijas) de un único programa shell.

Cuando una shell padre hace fork, el proceso padre (la shell padre), transfiere mucha información al proceso hijo (subshell). Si la llamada que hace es exec, la transferencia no es necesaria porque esa información se mantiene para el código objeto ejecutable. La información que se transfiere o se mantiene es:

El Entorno: Lista de nombres de variables y sus valores asociados.

Directorio actual: Este directorio será utilizado como punto de comienzo para cuando se utilicen paths relativos.

Ficheros abiertos: Una tabla con los ficheros abiertos por el proceso padre, con el valor actual de los punteros asociados a esos ficheros que se están utilizando.

ENTRADA Y SALIDAS STANDARD

Los programas BNU/Linux (y la práctica totalidad e los programas UNIX), están escritos para manejar la entrada y salida standard. Esta peculiaridad consigue que los OS tipo UNIX, sean muy flexibles. La mayoría de los comandos GNU/Linux usan la entrada y las salidas standard. La forma en que las utilizan no siempre es la misma, algunos la usan por defecto y otros necesitan de opciones especiales. Para saber que hace cada comando con respecto a la entrada y las salidas standard, es necesario consultar el manual en línea de esos comandos (man). Es importante saber como gestionan los comandos la entrada y las salidas standard, a la hora de hacer un script. Hay comandos que no usan las salidas standard y por lo tanto es imposible (o muy complicado), entubarlos.

Los programas que utilizan la entrada y la salida standard, leen y escriben de tres descriptores de ficheros predefinidos que se asociarán con los dispositivos específicos en el momento de la ejecución. Los tres descriptores utilizados son:

0 Entrada Standard (stdin). Es el descriptor del fichero desde el que son leídos los datos de entrada que serán procesados por el programa. La entrada standard por defecto es el terminal desde el que se lanza el comando (el teclado).

1 Salida Standard (stdout). Es el descriptor del fichero donde el programa lanzado escribirá los datos de salida (bien para mostrarlos o para servir de entrada a otro programa). La salida standard por defecto es el terminal desde donde se lanza el comando (la pantalla).

2 Salida standard de errores (stderr). Es el descriptor del fichero donde el programa escribirá los mensajes de error generados durante la ejecución del programa. La salida standard de errores por defecto es el terminal desde el que se lanza el comando. (la pantalla).

Redireccionar es desviar la entrada o las salidas standard de los comandos a otros ficheros. Hay que tener en cuenta que en GNU/Linux "todo es un fichero", por lo que, cuando hablamos de terminales, consolas (shell), en realidad estamos hablando también de ficheros.

Redireccionamiento de entrada

Consiste en tomar la entrada del comando de otro dichero que no sea el terminal. Para hacer este tipo de redirección se utiliza el símbolo < (menor que).