CURSO DESDE 0 DE GNU/LINUX.
Versión 2.

1ª Recopilación
Entregas desde la 0 a la 40

Luis García Galván (Matados'2k)

Agradecimientos:
· Revisión del documento: karuchi (Carolina García).

Página oficial y dominio de mi propiedad http://matados2k.es
Matados'2k Usuario y moderador de foro.noticias3d.com
Matados'2k Usuario y moderador de www.sinuh.org
,
matados2k (arroba) gmail (punto) com

Este documento está sometido a la licencia de creative commons en su variante "Reconocimiento-NoComercial-SinObraDerivada2.5 España" . Es de agradecer que se comunique al autor el uso de este documento en otro medio y se debe incluir de forma **obligatoria** este recuadro y los agradecimientos.

Reconocimiento-NoComercial-SinObraDerivada 2.5 España

Usted es libre de:

- Copiar, distribuir y comunicar públicamente la obra

Bajo las condiciones siguientes:

- **Reconocimiento**. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).
- **No comercial**. No puede utilizar esta obra para fines comerciales.
- **Sin obras derivadas**. No se puede alterar, transformar o generar una obra derivada a partir de esta obra.

    - Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
    - Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor
    - Nada en esta licencia menoscaba o restringe los derechos morales del autor.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

1ª Recopilación
Entregas desde la 0 a la 40

Luis García Galván (Matados'2k)

Índice:

PRÓLOGO

Este recopilatorio nace del "Curso desde 0 de GNU/Linux" que se empezó a publicar el 11 mayo de 2004 en su primera versión en Sinuh, y actualmente es publicado en mi web o blog personalnac http://matados2k.es [La vida entre chip y chip] y replicado en http://sinuh.org [SINUH] y http://noticias3d.com [Noticias3d] como sitios oficiales.

El proyecto nació originalmente como ayuda a mi buen amigo Ricardo, ex-moderador de los foros de Noticias3d (Ricardo Alfa) para su proyecto que nunca puso en marcha de http://comprahardware.com.ar [Sólo esta registrado el dominio] para el cual escribí los tres primeros capítulos. Dado que el proyecto de Ricardo no arrancaba y que empecé a participar en Sinuh, me puse en contacto con ellos y más concretamente con Antonio Rodríguez Capita e Inés García Guillo (Autores de la "Guía de aprendizaje: Gnu/Linex") y se comenzó a publicar el curso. Aunque por desgracia, debido a mi cada vez menor tiempo libre y dejadez, cayó en el olvido el 15 junio del 2005 en su entrega número 30, que es donde terminó su primera versión.

Esta primera versión tuvo una buena acogida en internet y un relativo éxito, e incluso fue impresa en su mayor parte en papel en la extinta revista "Linux Free Magazine" de Kernel Produktion, a la cual dí permiso de publicación cuando cambié la licencia para que pudiera seguir publicándolo.

El abandono del curso siemprese me quedó como una espinita clavada, y después de mi abandonado intento de blog en http://extreblog.com (el curso no fue lo único que dejé abandonado) y mi puesta en marcha de mi blog actual, unido a los continuos ánimos de Carolina García (Karuchi), revisora, incondicional del curso y futura esposa, decidí volver a ponerlo en marcha.

Primero empecé publicando la corrección y revisión de las entregas existentes el 6 Noviembre del 2006, y luego continué escribiendo más entregas. A día de hoy, salvo raras excepciones, escribo una entrega a la semana. Pero es más, ahora también publico semanalmente el "Curso desde 0 de programación C bajo GNU/Linux", del cual espero que llegue el momento de publicar otra recopilación como esta.

Así que espero que saquéis provecho de la recopilación, aunque barajo la idea de poder reutilizar todo este material y ampliarlo para hacer un libro como Dios manda, ¿Quién sabe?

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 0. Uno, dos... probando, sí, hola... uno, dos.

```
$ make fire
Make: Don't know how to make fire. Stop.
```

Presentación.

A la tercera va la vencida, después de varios intentos de revisión y continuos abandonos del curso, tomo aire y cojo lo que un día comencé, espero esta vez sí darle más continuidad salvo que el trabajo no me lo permita, que fue uno de los motivos de mi abandono anterior.

Para los nuevos: soy Matados'2k el desaparecido, muchos ya me conocen de los foros de noticias3d.com, otros me conocen por www.sinuh.org por donde no aparezco desde hace muuuucho tiempo y creo que me matarán XD por no cumplir lo que les prometí y otros tantos me conocerán por la extinta Linux Free Magazine que publicaba este curso.

A los nuevos, presentarme y a los no tan nuevos, mandarles un saludo y en general agradecer todos esos comentarios de apoyo recibidos allí donde se publicaba el curso, comentarios que me llenan de satisfacción al leerlos y me daban muchos ánimos para seguir. Por supuesto, gracias por todas la críticas buenas y malas y correcciones al curso, mi punto de vista no tiene por qué ser exacto ni yo perfecto, gracias siempre por vuestra comprensión y ganas de ayudarme a corregirlo. En esta edición serán corregidas todas las erratas.

Sólo me queda dar gracias en especial a mi novia, amiga y futura esposa Karuchi, correctora y sufridora de la publicación de este curso, dar la gracias a sinuh por su apoyo, a la comunidad de foro.noticias3d.com donde he aprendido mucho más de lo que yo aquí puedo enseñar y donde he conseguido un buen puñado de amigos.

Mi época de estudiante ya pasó, mi época de becario terminó y ahora me encuentro en el mundo laboral, soy Ingeniero en Informática actualmente metido en el mundo de los sistemas de información geográfica donde en buena parte me muevo por software libre, aunque no siempre es así.

Soy un convencido de la filosofía del software libre y usuario habitual de GNU/Linux. Y bueno, esta es una aportación que quiero hacer a la comunidad que tanto me ha dado, sí es pequeña ya lo sé, pero espero que sea útil, algún día espero poder aportar una gran contribución, ideas no me faltan pero de tiempo no dispongo mucho.

¿Por qué escribir un curso de GNU/Linux? Pues sencillamente por satisfacción propia.

¿Cuál será la duración? Pues de momento será indefinida ya que no creo que los temas a tratar se nos acaben, así que paciencia que hay mucho camino.

¿Cada cuánto escribiré una entrega? Pues he pensado que en esta edición cada quince días pondré una, todo depende de mi tiempo disponible y de lo que me animéis vosotros (ya sabéis que es muy aburrido predicar en el desierto).

¿Dónde encontrarme? En los foros de noticias3d.com, en los foros de www.sinuh.org y por fin en mi propia web http://matados2k.es. Rogaría que todas las preguntas que salgan sobre el curso me las hicierais en los foros (evidentemente no hace falta decir la sección) y evitéis añadirme al messenger porque no me gusta que me añadan para freírme a preguntas (las preguntas a los foros) y puede que si

abusáis os bloquee. Si queréis podéis mandarme e-mails, pero si alguno ya me conoce bien sabe que soy muy perro y puede que tarde en contestar (sin embargo postear no me da nada de pereza) pero contestar, contesto (sugerencias, criticas, comentarios .. nada de preguntas).

Por dónde empezar.

Bueno, a algunos puede que no les guste por dónde voy a empezar pero es que yo pienso que es la parte más importante y la que más chicha tiene y es la consola de texto. Algunos de vosotros la veréis como algo anticuado y complicado (nada más lejos de la realidad), pero sinceramente ya os daréis cuenta de que la consola de comandos da una libertad muy grande y es muy potente, además de necesaria, que después nos pasa lo que nos pasa. Sí, tocaremos lo que son los entornos gráficos, pero cuando tengamos visto lo importante, que en modo gráfico también hay muchas cosas interesantes de ver además de ser lo que más vais a usar.

Dónde se publicará oficialmente el curso.

La web oficial del curso es http://matados2k.es, este será el primer sitio donde será publicado, con posterioridad se publicará una réplica entre 2 y 3 días más tarde en la web de http://www.sinuh.org y si es posible en la web de http://www.noticias3d.com , estos y sólo estos serán los sitios donde yo oficialmente lo publicaré, del resto de sitios donde se replique manteniendo la licencia me desentenderé ya que para eso es un curso de libre distribución, no comercial, con reconocimiento y sin obra derivada, lo único que pido es que se respeten las condiciones.

Finalizando

Bueno, pues después de esta entrega tostón deciros que empezaremos la 1ª entrega con los conocimientos básicos necesarios para instalar una distribución y viendo cómo es nuestro sistema operativo y cómo está estructurado para entenderlo mejor

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 1. Manos a la obra (1ª parte).

```
Si  Atila y su caballo  hubieran usado Linux...
$ for i in spqr/*; do rm -rf $i; cut grass; done
```

Para empezar

Lo aquí expuesto ha sido corregido y ampliado con los comentarios que ensu día se hicieron. No voy a extenderme más de lo que en su día me extendí, ya que el texto está pensado para dar una mínima pero útil visión de que es en concreto GNU/Linux y el software libre, sin entrar en detalles ni las discusiones técnicas tan habituales en los ya avanzados usuarios del software libre, por dos motivos: se escapa del objetivo de este curso y, segundo, no veo útil abrumar con información y discusiones al lector con menos conocimientos al que va dirigido este curso.

Mitos y leyendas

Bueno, para empezar y antes de todo, lo primero que quiero es desmitificar afirmaciones que mucha gente tiene como ciertas.

El software libre es gratuito. Esta es la creencia más arraigada entre la gente y por supuesto totalmente falsa. El software libre no tiene por qué ser gratuito, aun cuando puedas conseguirlo gratuitamente. Es perfectamente normal cobrar por ello (si no mirad la cantidad de distribuciones que son comerciales), generalmente no hay un pago por licencia, siendo lo normal el pago por el soporte y mantenimiento, que un usuario normal no suele usar pero sí generalmente empresas. Ni que decir tiene que hay muchas más formas, pero no es el objetivo de este curso, al menos de momento.

Linux sólo es para hackers y programadores. Otra cosa que es totalmente falsa y que aún sigue en la mente de muchos. Linux es cada vez más un sistema operativo de lo más versátil al que le queda muy pocos campos en los que se queda cojo.

Linux es difícil. Realmente esto con los nuevos escritorios y las instalaciones tan sencillas que se nos ofrecen es ya algo del pasado. Lo único que requiere Linux es tener unos conocimientos mínimos que tampoco son nada del otro mundo para poder empezar a usarlo. Otra cosa es que no tengamos paciencia ya que el cambio siempre es costoso, tanto por la comodidad de lo conocido como por querer aprender algo nuevo en poco tiempo.

Linux es seguro. Esto es según nosotros queramos que lo sea, por defecto sí es más seguro, pero esto no da más que una falsa seguridad. Hay que recordar que no hay configuraciones por defecto buenas y un sistema mal mantenido y configurado es algo bastante vulnerable. Linux es tan seguro como nosotros de paranoicos para que lo sea.

En Linux no existen virus. Otra cosa totalmente falsa y que muchísima gente cree o piensa que hay 10 ó 12 a lo sumo. Hay varios factores por los que los virus en Linux no tienen mucho alcance. Lo primero es que el usuario medio de Linux no es ni mucho menos el usuario medio de Windows, y muchas técnicas de ingeniería social usadas por los virus no surten mucho efecto, amén de que se acostumbra a conocer el origen de los programas. Otro factor a tener en cuenta es que Linux no es la plataforma mayoritaria, y bien es sabido que todo creador quiere que su creación llegue lo más lejos posible. Y totalmente obvio es que la seguridad es mejor y los errores se corrigen mucho más rápidamente al ser un sistema abierto. Para los incrédulos, decir que hay más de 300 virus para Linux, pero lo bueno es que muchos de ellos son pruebas y virus de "laboratorio". Lo que sí podemos es

llegar a la conclusión de que el riesgo de virus en un sistema Linux es tan pequeño que no es nada preocupante no tener un antivirus, que generalmente sirven para limpiar virus de Windows ya que hay muchos servidores Linux en Internet.

Qué puedo y no puedo hacer con software libre.

Los derechos que tenemos con el software libre (la licencia GPL de la GNU es la más representativa aunque existen muchísimas más como por ejemplo la BSD, ver http://www.fsf.org/licensing/licenses/index_html#GPLCompatibleLicenses) es que podemos copiarlo, modificarlo e incluso redistribuirlo y cobrar por ello si quisiéramos (otra cosa es que nos paguen por ello), pero bajo unos deberes. En el caso concreto de nuestro representativo ejemplo, la licencia GPL, tienes que distribuir el código, respetar la licencia y las notas de autor, siendo ilegal apropiarse de ese código. Si tú modificas algo con licencia GPL el nuevo código será forzosamente GPL (otra cosa es que tú uses código propietario apoyado en GPL), y evidentemente tú serás el autor de ese código que añades. Para una mejor información lee la propia GPL, es un poco pesada pero bueno.

Nota: Software abierto no es lo mismo que software libre, con un software abierto puedes ver su código pero no tienen que cumplirselos otros derechos.

Características de GNU/Linux.

Multitarea. Puede hacer varias cosas a la vez y además hay que destacar que es el multitarea más eficiente que he visto nunca.

Multiusuario. Cualquier Linux puede tener muchos usuarios concurrentemente usánddo en **Tiempo real y sin ningún programa añadido**, ya sea por consolas virtuales, sesiones remotas o terminales.

Multiprocesador. Maneja dos o más procesadores a la vez, sin por ello tener que pagar nada de más.

Soportado por múltiples arquitecturas. ¿Conoces alguna arquitectura de computador? Seguro que ya hay una versión de Linux para ella a menos que sea demasiado nueva (y ya tiene que serlo) para que no la haya. Hay que destacar que no existen versiones de Linux para arquitecturas Intel más bajas que el 386. ¿Sale una nueva arquitectura? Seguro que si no está ya soportada pronto lo estará, mirad si no qué rápido fue soportado el AMD 64.

Dónde conseguir GNU/Linux.

Pues es bien sencillo, si no conoces a nadie que tenga una distribución que pueda pasarte, puedes conseguirlas en páginas como www.linuxiso.org donde encontrarás casi todas las distribuciones disponibles o en ftps como el de ftp.rediris.es/pub. Si no tienes una buena conexión a Internet u oportunidad de que alguien baje una por ti, sólo tienes que acercarte a una librería o kiosco y buscar una revista especializada, seguro que alguna trae una distribución incluida.

Para empezar yo os recomendaría usar distribuciones como Red hat, Suse o Mandrake. Personalmente yo me decanto por Mandrake para los más novatos, pero está claro que es cuestión de gustos. Una vez dominado un poco el tema es entonces cuando aconsejaría el uso de una Debian o Gentoo.

También si somos algo perros ono nos interesa tenerla en nuestro disco duro o simplemente por miedo, existen versiones Live (que arrancan de CD y funcionan en memoria sin tener que instalar nada), en este caso me decanto indiscutiblemente por knoppix (basada en Debian), existiendo muchas otras como Suse Live, Linex, Guadalinex (Estas dos últimas tienen su correspondiente versión instalable y pertenecen a la Junta de Extremadura y a la Junta de Andalucía respectivamente, ambas

basadas en Debian), Linuxin… amén de muchas más.

Otras distribuciones recomendables pueden ser Lyrcoris, Yoper, Slackware (esta es otra de las grandes que con el tiempo ha ido perdiendo adeptos, pero sin duda Slackware es muy buena), LambdaUX y desaconsejaría distribuciones con Lindows OS.

Pero como siempre todo es cuestión de gustos, y lo aquí expuesto esta basado en la experiencia y mi opinión personal. Seguro que encuentras el sabor Linux (Linux like) que más te gusta.

Instalar GNU/Linux

Evidentemente me es imposible hacer una guía de instalación para cada una de las distribuciones aquí expuestas y tampoco voy a explicar una o dos. Simplemente voy a dar unos conocimientos previos necesarios para poder desenvolverse bien en casi todas las distribuciones (explicar para una Debian o un Gentoo, por ejemplo, es más complicado) ya que muchas de ellas tienen instaladores casi tan simples o más aún que cualquier Windows.

¡Ojo, no enseño las herramientas, sino que os inicio en los conocimientos necesarios para que utilices las herramientas que creáis oportunas!

Lo primero que debemos saber es cómo van las particiones en los discos duros. Un disco duro puede tener hasta 4 particiones de tipo primario y extendido. Sólo puede haber una partición extendida, con lo que podemos tener de 1 a 4 primarias o de 1 a 3 primarias y por último una extendida. Al menos tiene que existir una primaria, y si hay varias primarias una de ellas tiene que ser la "activa", que es la que arrancará el ordenador (mbr, que sólo las primarias tienen). Hay que decir que las extendidas no se usan físicamente y sólo sirven para definir dentro de ellas lo que denominamos particiones lógicas. Dentro de una partición extendida podemos definir todas las lógicas que queramos. Esto puede parecer un trabalenguas, si así te parece, lee este párrafo varias veces y haz un diagrama.

Lo más normal y lógico, desde mi punto de vista y experiencia, es tener una primaria y una extendida, y dentro de ésta las lógicas que vayamos a necesitar. Tanto Linux como otros sistemas operativos pueden funcionar perfectamente en particiones lógicas sin problemas gracias a los gestores de arranque (cuando no existían éstos, evidentemente el limite de s.o. por ordenador era de 4 y había que andar activando y desactivando las particiones, por supuesto primarias, con lo que las lógicas quedaban para datos).

Bien, ahora Linux para instalarse necesita como mínimo 2 particiones: una **swap** y otra '/' (llamada raíz). Linux utiliza por defecto (por rapidez y eficiencia) una partición en vez de un fichero de intercambio (aunque realmente sí puede manejarlos), para los que no sepan qué es un fichero de intercambio sólo comentar que cuando el ordenador necesita más RAM que la que físicamente tiene simula tener más usando espacio del disco duro. El tamaño de ésta, por regla general, suele ser el doble de la RAM que tiene nuestro ordenador, yo no aconsejo nunca pasar de 512 (por lo menos actualmente) ya que si seguimos la regla y tenemos 512Mb por ejemplo, tener un fichero de intercambio de 1G es desde mi punto de vista desperdiciar espacio. Cabe destacar que esta regla esta desfasada pero nos puede servir de guía.

Ahora bien, si sólo queremos instalar Linux necesitaremos como mínimo una primaria y una swap (mejor que sea lógica). Pero como éste no es el caso de mucho de nosotros, que queremos tener otros sistemas operativos, típicamente Windows, lo que debemos hacer es instalar los s.o. desde el más antiguo al más moderno y por último instalar Linux.

Ejemplo 1: Queremos Windows 98 y Linux, o mejor aún, cualquier Windows y Linux.

Primaria: Windows
Extendida:
  Lógica 1: ‘/’
  Lógica 2: swap

Ejemplo 2: Queremos Windows 98, Windows XP y Linux.

Primaria: Windows 98
Extendida:
  Lógica 1: Windows XP
  Lógica 2: ‘/’
  Lógica 3: swap

Espero que con estos 2 ejemplos quede claro.

Ahora, como muchos sabéis cada partición tiene un formato. El formato típico de los Windows es FAT (FAT 16 para MS-DOS y Win95, FAT 32 para Win9x/ME/XP home, NTFS para Win NT/200X/XP y FAT 15 para disquetes), pues bien el estándar de Linux es EXT3 (Actualmente se usa EXT3 y EXT2 en su defecto, EXT como tal está en desuso), pero puede usar casi cualquier tipo de formato (creo que más de 80). Otro formato muy usado en Linux y muy común es RaiserFS. Así que, como se supone que si estás leyendo esto eres aún principiante, pon la partición ‘/’ como ext3 o en su defecto ext2 (Nota: La swap es un formato en sí misma). Ni que decir tiene que la swap sólo la ve y la usa el sistema operativo.

Siento cortar en este momento, pero el espacio de esta entrega se acaba así que en la próxima entrega continuamos.

Entrega 2. Manos a la obra (2ª parte).

```
% ping elvis.rice.edu | awk '{print substr($1,1,5), $2, $3}'
elvis is alive
```

Instalar GNU/Linux (continuación)

En un principio no debemos tener mucho miedo a la hora de quedarnos cortos con la swap, ya que Linux puede manejar hasta 8 ficheros de intercambio y/o swaps (a menos que haya aumentado el número con la versión 2.6 del Kernel o sea Linux, hay que recordar que Linux es solamente un Kernel o núcleo de sistema) y siempre pueden definirse más a posteriori.

Con lo que el segundo ejemplo nos quedaría:

Primaria: Windows 98 Formato: FAT 32 (también funciona sobre FAT 16)
Extendida:
  Lógica 1: Windows XP Formato: NTFS (también funciona con FAT 32)
  Lógica 2: '/' Formato: EXT3 o EXT2 (Opcionalmente RaiserFS y más...)
  Lógica 3: swap Formato: SWAP

Bueno ahora sólo queda ¿con qué hacerlo? Pues con lo que más rabia te dé, tanto con un Partition Magic como con los distintos programas libres para ello, incluso con un CD de instalación de Linux puede hacerse (cosa que yo he hecho muchas veces). De todas formas, si vas a instalarlo todo desde 0 puedes hacer lo siguiente (y esto sólo es una forma de las muchas que se puede hacer): haz todas las particiones con el fdisk de Windows (esto está claro para la gente que quiere empezar a migrar y no quiere complicarse, no os tiréis encima mía, linuxeros, que os conozco) con particiones FAT32 y luego según vas instalando sistemas operativos vas cambiando los formatos, ya que todos dan esa opción.

También existe la posibilidad de reparticionar el disco duro quitando espacio a particiones para crear otras nuevas. Para ello hay que desfragmentar y mandarlo todo al comienzo de la partición, para lo que desaconsejo el de Windows en cualquiera de sus versiones, ya que suelen dejar muchas veces cosas por medio (¿casualidad?, no lo creo), para después con los citados programas anteriores redimensionar unidades (incluso los de las instalaciones de Linux) y crear nuevas. Ni que decir tiene que hagáis copias de seguridad "por si acaso".

Otra cosa que seguramente necesitarás es saber cómo llama Linux a los discos duros. Bueno, como ya explicaré mejor más tarde en linux todo cuelga de '/', nada de unidades C:, D:, E:, y el lugar donde Linux tiene los dispositivos es en /dev (device).

El formato de los dispositivos **IDE** es el siguiente: hdXY donde X puede ser 'a', 'b', 'c' o 'd', y donde 'a' es maestro del primer bus ide, 'b' el esclavo, 'c' el maestro del segundo bus ide y 'd' el esclavo. Y es un número, del 1 al 4 son las particiones primarias y del 5 en adelante las lógicas.

Por lo tanto, si por ejemplo tenemos dos discos duros en el primer ide y en el segundo ide una grabadora y un dvd, estando el primer disco duro particionado como en el ejemplo 2, la forma de nombrarlos sería:

Primer disco (maestro, ide 0) /dev/hda (Nos referíamos a todo el disco como tal)

Partición con win 98 se (primaria) /dev/hda1
Partición con Win XP (lógica) /dev/hda5
Partición con Ext3 (lógica) /dev/hda6
Partición Swap (lógica) /dev/hda7
Segundo disco duro (esclavo,ide 0) /dev/hdb
Partición de datos (primaria) /dev/hdb1
Grabadora (maestro, ide 1) /dev/hdc
Dvd (esclavo, ide 1) /dev/hdd
(Nótese que lo que no son discos duros no necesitan número)
Espero que con esto ya quede claro.

Sin embargo, para los dispositivos **SCSI** y actualmente los discos **SATA** y otros dispositivos como los **PENDRIVES** se cambia la 'hd' por 'sd'. Las disqueteras son fdX, donde X es el número de la disquetera empezando por 0.

Aquí dejo una tabla aclaratoria:

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, partición primaria 1	/dev/sda1
Segundo disco duro SCSI (todo el disco)	/dev/sdb
Segundo disco duro SCSI, partición primaria 1	/dev/sdb1

Bueno, y como último dato que quizás podáis necesitar son los dispositivos del ratón:

/dev/psaux Si el ratón esta en el puerto PS/2
/dev/ttyS0 Si el ratón esta en el primer puerto serie
/dev/ttyS1 Si el ratón esta en el segundo puerto serie
/dev/sdbX Si el ratón esta en el puerto usb X

Otra cosa que os van a pedir es una clave de root, el root es el usuario con más privilegios de Linux o sea el administrador, usa una buena clave y no la uses nada más que para cosas que no puedas hacer con un usuario normal. Seguramente, acto seguido te pedirán que introduzcas los usuarios, así que hazte una cuenta normal para ti y otra para cada usuario que vaya a usar el sistema.

Bueno, espero que con estos pequeños conocimientos no tengáis muchas dificultades para instalar la distribución de Linux que más os guste, y si no te gusta ninguna quién sabe si algún día tendrás la tuya propia.

¡¡¡Trata de arrancarlo, Carlos!!!!....

Por fin tenemos ya instalada nuestra distribución ya sea la favorita, la recomendada o por casualidad. Arrancamos nuestro ordenador, bip, chequeo, lilo o grub (o nada para los que tengan sólo Linux), elegimos que arranque Linux y según hayamos instalado se nos presenta una bonita pantalla gráfica pidiéndonos el nombre de usuario y clave, o bien nos arranca directamente en la consola. Los que ya

estén en la consola bien, porque es aquí donde empieza nuestro curso, y para los que no pueden hacer dos cosas: entrar y arrancar una consola virtual (enredad y poquito que seguro la encontráis, buscad por consola o terminal) o bien vais a un terminal de la siguiente forma *: Ctrl+Alt+Tecla de función.

Donde Tecla de función es la tecla de función (para los que no se enteran F1, F2, F3…) que coincide con el número del terminal virtual al cual queremos acceder.

* Antes de nada indicaros que en Linux pueden existir más de un terminal, en los que funciona una consola (de hecho la interfaz gráfica se muestra en un terminal en el que está corriendo xfree86 o XOrg, que son los motores gráficos de Linux). Como físicamente sólo puede haber un terminal en una pantalla, se usan los llamados terminales virtuales, normalmente vienen por defecto de 4 a 6 terminales y una última en la que se está ejecutando el modo gráfico si elegiste arrancar en este modo (normalmente el 7).

Pues nada, entramos en nuestro sistema con el usuario que ya dimos de alta (mejor que no uséis root a menos que sea necesario, por tonto que esto os parezca).

Qué nos encontramos

Para empezar hay que decir que todo en Linux son ficheros, los directorios incluidos, que no son más que ficheros con enlaces o información a más ficheros (no asustarse que no hay que ir mirándolos, el uso es como en cualquier otro sistema), incluso los dispositivos son ficheros un tanto especiales, pero que al fin y al cabo están representados en el sistema como ficheros (acordaros de cómo se llaman las particiones).

Como ya comenté antes aquí no existe A:, B:, C:, D: … ni nada parecido, en cualquier sistema Unix/Linux todo está a partir de lo que denominamos raíz y se representa mediante el símbolo '/', a partir de aquí cuelga todo lo demás formando un árbol (se le denomina árbol a toda la estructura de directorios y ficheros). A los datos de una partición no se accede a través de /dev/hdXY como algunos hayan podido deducir, ya que esto representa el dispositivo en sí, no a su contenido. Pues bien, las particiones en Linux hay que montarlas y se pueden montar en la parte que queramos, la que más rabia nos dé, y simplemente hay que acceder al directorio donde esté montada la partición o disco duro para ver el contenido de éste. Esto pasa absolutamente con todo, incluso si tenemos que usar carpetas compartidas en la red, primero se montan y luego se acceden como si de tu propio disco duro de se trataran.

Una vez visto esto, vamos a ver cómo están distribuidas las cosas dentro de nuestro sistema, cabe destacar que puede haber variaciones de unos sistemas a otros pero en general todos siguen las mismas reglas y suelen seguir el estándar "FSH" (Filesystem Hierarchy Standard):

/ Directorio raíz, de aquí cuelga todo y es el principio de nuestro sistema de ficheros.

/bin De binarios, en este directorio se encuentran los programas esenciales para el sistema.

/dev De device, en este directorio nos encontramos todos nuestros disposivos, tales como discos duros, terminales, consolas o terminales virtuales, buses…

/etc Contiene los ficheros de configuración del sistema y sus subdirectorios los ficheros de configuración de lo que representen.

/sbin Como /bin pero sólo los esenciales para el sistema que usa el súper usuario, o

sea root. Así que normalmente no tendremos acceso a esta carpeta y sus ejecutables.

/home — Contiene los 'home' de cada usuario, o sea las carpetas con los ficheros de cada usuario. Por hacer analogías es como la carpeta "Documents and Settings" de cualquier Windows 2000/XP

/lib — Contiene las imágenes de las librerías compartidas del sistema. Estos ficheros contienen código que es usado por multitud de programas.

/proc — Es un sistema de archivos virtual, y los archivos que contiene residen en memoria. Aquí nos encontramos representados los procesos que corren en nuestro sistema y algunos "especiales". Intenta encontrar algo parecido en Windows >:D.

/tmp — Pues donde va a parar la basura que genera el sistema, o sea, los malditos temporales que generan nuestros programas.

/usr — Es un directorio realmente importante, aquí se encuentran las cosas que no son esenciales en el sistema y dentro de él tenemos otro bin, sbin, etc, lib,…

/usr/include — Incluye los ficheros de cabecera para el compilador de C y varios más.

/usr/local — Igual que /usr, pero los específicos de tu sistema y que probablemente no estén en otro Unix o Linux.

/usr/src — Es un directorio donde se encuentran los códigos fuente del sistema. Por ejemplo /usr/src/linux contiene el fuente del Kernel

/var — Contiene directorios que a menudo tienden a crecer y cambiar su tamaño, aquí solemos encontrar cosas como las paginas que aloja un servidor en/var/www.

/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/mail hasta que lo lea o lo borre. Artículos nuevos de las "news" tanto salientes como entrantes pueden encontrarseen /var/spool/news, etc.

/opt — De optional. Como curiosidad, si instaláis el Messenger de Yahoo y no lo tenéis se os creará y se instalará allí.

/mnt — Aquí es donde normalmente montamos discos duros, carpetas en red, cd´s, disquetes…

Bueno, y como el espacio de esta entrega se acaba, comenzaremos con los primeros comandos en la próxima entrega donde aprenderemos, entre otras cosas, cómo movernos por este árbol de directorios.

Entrega 3. Moviéndonos por el sistema.

```
% man: why did you get a divorce?
man:: Too many arguments.
```

El intérprete de comandos

Bueno, pues lo primero que tenemos a la vista es el intérprete de comandos esperándonos para recibir órdenes. El intérprete de comandos es para Unix/Linux como el COMMAND.COM del MS-DOS sólo que mucho más potente y además no hay un solo intérprete sino que hay varios. El más usado y que probablemente estés usando es el bash, el nombre bash proviene de BourneAgain SHell, pero existen otros como el sh (el segundo más usado) o tsh. Para cambiar de uno a otro sólo tienes que teclear su nombre y para volver al anterior teclear exit.

Las diferencias que hay entre los distintos intérpretes no nos preocupan ahora, se diferencian fundamentalmente a la hora de hacer shell scripts pero eso lo veremos más adelante.

Y qué es lo primero que nos encontramos: el **prompt**, en mi caso es algo tal que así:

```
[matados2k@fortaleza matados2k]$
```

Como su nombre nos sugiere (promtpt significa solicitud), el intérprete de comando nos está solicitando que le demos una orden. Como podemos observar el prompt nos da una información, ésta depende de cómo lo tengamos configurado, pero de esto hablaremos más adelante en el curso. En mi caso lo primero es el nombre del usuario seguido del nombre de mi máquina, lo siguiente es el directorio donde me encuentro, en este caso estoy en /home/matados2k .Y por último el símbolo '$' indica que soy un usuario normal, si tuviera un '#" indicaría que soy superusuario (root), como en el siguiente ejemplo:

```
[root@fortaleza home]#
```

Una cosa hay que quedar clara, los sistemas Unix/Linux son **case sensitive,** o sea, que diferencian entre minúsculas y mayúsculas, para el ordenador no será lo mismo Ls, lS, ls, LS, ya que para el sistema operativo son todas diferentes.

Moviéndonos por los directorios

La orden para movernos por la estructura de la red es **'cd'** y su formato es el siguiente:

cd [directorio]

Donde 'cd' es el comando y lo que está en corchetes es el argumento, al estar en corchetes es opcional.

Para movernos de un directorio a otro usamos este comando, tenemos 2 formas de nombrar un directorio: la primera es por su ruta absoluta y la segunda es por su ruta relativa. La absoluta se refiere al nombre completo empezando desde '/', un ejemplo seria ir a /etc (yo estoy en /home/matados2k).

```
[matados2k@fortaleza matados2k]$ cd /etc
[matados2k@fortaleza etc]$
```

Esto tiene un problema ya, que siempre tenemos que escribir la ruta completa a donde queramos ir. Para no tener que hacerlo usamos las rutas relativas: ponemos el nombre del directorio que queremos entrar pero a partir desde el que estamos. Para ver esto volveremos a /home/matados2k (en vuestro caso el directorio de trabajo vuestro) y de allí iremos a /home/matados2k/Documents.

```
[matados2k@fortaleza etc]$ cd /home/matados2k
[matados2k@fortaleza matados2k]$ cd Documents
[matados2k@fortaleza Documents]$
```

En el primer caso hemos usado de nuevo una ruta absoluta y en el segundo como ya estábamos en /home/matados2k hemos puesto sdo Documents para entrar en /home/matados2k/Documents, que seria su ruta absoluta.

Dentro de todo directorio existen dos directorios especiales que son '**.**' y '**..**'. El primero hace referencia al directorio actual, es decir, si haces 'cd .' te quedas donde estás (el directorio especial '.'ya veréis más adelante lo útil que es), y el segundo hace referencia al directorio padre, o sea, si estamos en /home/matados2k y hacemos 'cd ..' terminaremos en /home.

```
[matados2k@fortaleza matados2k]$ cd .
[matados2k@fortaleza matados2k]$ cd ..
[matados2k@fortaleza home]$
```

Estos dos directorios especiales los usaremos en la rutas relativas, es lo mismo hacer 'cd ./Documents' que 'cd Documents' que para este comando en concreto nos da lo mismo pero en otros puede que necesitéis usar './'.Imaginaros que desde /home/matados2k queréis entrar (si tuvierais otro usuario llamado así) en /home/kronem, para no escribir la ruta absoluta recurriremos a 'cd ../kronem' y conseguimos mediante una ruta relativa dar un salto atrás en el árbol y entrar en 'kronem'

```
[matados2k@fortaleza matados2k]$ cd Documents
[matados2k@fortaleza Documents]$ cd ..
[matados2k@fortaleza matados2k]$ cd ./Documents
[matados2k@fortaleza Documents]$ cd ..
[matados2k@fortaleza matados2k]$ cd ../kronem
[matados2k@fortaleza kronem]$
```

Bueno pues no se vayan todavía, aun hay más. Hay una referencia especial, estemos donde estemos si hacemos 'cd ~ ' volvemos a nuestro directorio de trabajo yaque ~ (bajo consola lo conseguimos pulsando la tecla Alt Gr +4) es una referencia a nuestro directorio de trabajo.

```
[matados2k@fortaleza matados2k]$ cd /usr/bin
[matados2k@fortaleza bin]$ cd ~
[matados2k@fortaleza matados2k]$
```

Pero... es que aún lo podemos hacermas fácil esta funcionadiad ya que 'cd ~' es lo mismo que escribir simplemente 'cd', haced la prueba. Y para el remate de los tomates una utilidad mas 'cd -' que nos devuelve al directorio anterior, no me refiero al padre sino al anterior donde nos encontrábamos.

```
[matados2k@fortaleza]$ cd /etc
[matados2k@fortaleza etc]$ cd –
/home/matados2k
[matados2k@fortaleza matados2k]$
```

Con esto ya hemos visto este comando tan simple pero útil.

Me he perdido ¿Dónde estoy?

Si nos perdemos, bien por que hemos enredado mucho moviéndonos por el árbol de directorios o bien porque nuestro prompt no nos indca en cada momento dónde estamos, tenemos un comando bien sencillo:

pwd

```
[matados2k@fortaleza matados2k]$ pwd
/home/matados2k
[matados2k@fortaleza matados2k]$
```

Antes de continuar...

Antes de continuar hay que comentar una cosa, para evitar confusiones explicaré un poco los 'alias'. El intérprete de comandos nos la opción de definirnos lo que denominaremos 'alias', por ejemplo, puedes hacer que la palabra 'casa' ejecute 'cd~', parece una bonita forma de crear comandos a nuestro gusto. Algunos de los comandos que aquí explicaré tienen definidos alias en muchas distribuciones por lo que el resultado puede que no sea el mismo. Si esto os ocurre teclead 'alias COMANDO_QUE_FALLA=COMANDO_QUE_FALLA' (ojo, sin espacios en el igual), por ejemplo, si cuando hicierais 'cd' os mandara a /usr/bin (es un ejemplo raro pero bueno) bastaría con ejecutar 'alias cd=cd'.

Para ver qué alias tenéis definidos ejecutad 'alias -p'

```
[matados2k@fortaleza matados2k]$ alias -p
alias l.='ls -d .* --color=tty'
alias ll='ls -l --color=tty'
alias ls='ls --color=tty'
alias mc='. /usr/share/mc/bin/mc-wrapper.sh'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-
alias --show-dot --show-tilde'
[matados2k@fortaleza matados2k]$
```

Lo que veremos a continuación es muy probable que tengas que hacerlo para que salga lo mismo que te estoy explicando. El uso de alias sobre todo se utiliza para ejecutar por defecto opciones muy útiles sin tener que escribirlas siempre, incluso si vienes del mundo de MS-DOS puedes definir todos tus comandos de siempre con los análogos en Linux, útil, ¿verdad? ;) (aunque esto último no lo recomiendo, no es bueno coger malos vicios XD).

Nota: los alias que definamos se esfumarán cuando cerremos nuestra sesión o reiniciemos la máquina, pero no tengáis miedo. Ya explicaremos más adelante cómo definir las cosas permanentemente.

Pues a mí me pica la curiosidad

Bueno, ya sabemos movernos, pero esto no nos sirve de mucho si no sabemos qué hay en nuestros directorios. Para eso tenemos un buen comando:

ls [opciones] [directorio]

Ahora ejecutaremos 'ls' tal cual y nos encontraremos con la siguiente salida:

```
[matados2k@fortaleza Documents]$ alias ls=ls
[matados2k@fortaleza Documents]$ ls
10350-1.jpg         Epsn0030.jpg                woman
9476-sparkle7.png   kdetutorial-1.1.2.tar.bz2   womanserv
[matados2k@fortaleza Documents]$
```

Nos encontramos con una salida algo pobre, porque en algunos casos no sabremos ni diferenciar qué es cada cosa: ¿un fichero? ¿un directorio? ¿en woman hay titis picantonas? (por cierto, la respuesta es no… ohhhhhhhh… es el nombre de mi proyecto de fin de carrera). Con la opción '-F' obtendremos más información:

```
[matados2k@fortaleza Documents]$ ls -F
10350-1.jpg         Epsn0030.jpg                woman/
9476-sparkle7.png   kdetutorial-1.1.2.tar.bz2   womanserv/
[matados2k@fortaleza Documents]$
```

Ahora podemos apreciar que a los directorios al final le añaden '/'. A ls también le podemos indicar el directorio que queremos mirar:

```
[matados2k@fortaleza Documents]$ ls -F /usr/local/bin
fbgnuboy*  romfixer*  sdlgnuboy*  supertux*  tuxnes*
xgnuboy*  zsnes*
[matados2k@fortaleza Documents]$
```

En este caso además vemos un * al final de estos ficheros, lo que nos indica que son ejecutables. También podemos encontrarnos conuna @ indicando que es un enlace. Otra opción interesante y que nos será útil es '-a' que nos muestra los ficheros (acuérdate de que en Linux todo son ficheros) ocultos.

```
[matados2k@fortaleza Documents]$ ls -F -a
./   10350-1.jpg        Epsn0030.jpg  kdetutorial-
1.1.2.tar.bz2  womanserv/
../  9476-sparkle7.png  .hola.txt     woman/
[matados2k@fortaleza Documents]$
```

Podemos observar que aparecen más ficheros que estaba ocultos en Linux, cualquier fichero que empiece por '.' es un fichero oculto contando los directorios especiales '.' y '..' que ya os comenté antes. Una opción realmente útil y que realmente os interesará es 'l' :

```
[matados2k@fortaleza Documents]$ ls -Fa -l
total 2144
drwxrwxr-x    4 matados2k matados2k     4096 may 13 02:07 ./
drwxr-xr-x   79 matados2k matados2k     4096 may 13 02:04 ../
-rw-rw-r--    1 matados2k matados2k   342064 abr 15 00:25 10350-1.jpg
-rw-rw-r--    1 matados2k matados2k   223535 abr 15 00:27 9476-sparkle7.png
-rw-rw-r--    1 matados2k matados2k  1298657 abr 22 23:10 Epsn0030.jpg
-rw-rw-r--    1 matados2k matados2k        5 may 13 02:07 .hola.txt
-rw-rw-r--    1 matados2k matados2k   284654 abr 14 15:24 kdetutorial-1.1.2.tar.bz2
drwxrwxr-x    4 matados2k matados2k     4096 abr 15 22:04 woman/
drwxrwxr-x    5 matados2k matados2k     4096 abr 19 11:32 womanserv/
[matados2k@fortaleza Documents]$
```

Para empezar veréis que se pueden mezclar opciones y que '-l' nos da una gran cantidad de información. La primera indica el tipo de fichero y los permisos (por ejemplo, drwxrwxr-x), los permisos no los explicaré ahora por que no toca, pero sí comentar que la primera letra indica qué tipo de fichero es ('d' nos indica que es directorio, '-' que es un fichero normal, 'c' que es un dispositivo orientado a carácter, 'b' dispositivo orientado a bloque y 'l' indica que es un enlace). Lo siguiente nos indica: el número de enlaces que existen a él, el propietario, el grupo propietario, el tamaño en bytes, la fecha y el nombre.

Por último ya sólo enseñaros otra opción que os gustará muchísimo '--color':

```
[matados2k@fortaleza Documents]$ ls -color
10350-1.jpg         Epsn0030.jpg                woman
9476-sparkle7.png   kdetutorial-1.1.2.tar.bz2   womanserv
[matados2k@fortaleza Documents]$
```

La verdad se explica por ella sola, ¿no? No seguiremos con más opciones de 'ls' porque hay casi tantas como letras del abecedario, tanto en mayúsculas como en minúsculas, así que puedes incluso probar 'ls' con tu nombre y demostrar lo triste que es tu vida perdiendo el tiempo en semejantes tonterías.

Despedida y cierre

Ya para finalizar sólo indicaros que, si estáis en modo consola, para cerrar la sesión hay que teclear 'exit' o 'logout', para parar la máquina 'halt' y para reiniciarla pulsar 'ctrl+alt+sup'.

Esto es todo lo que ha dado de sí el moverse por el sistema, así que en la próxima entrega trataremos el tema del manejo de ficheros (copia, creación, renombre...).

Por cierto, ya sabía que probarías 'ls' con tu nombre :P

Entrega 4. Manejando ficheros.

> % make ' ' bang ' ' with gun
> make: Fatal error: Don't know how to make target ` '

Creando directorios

Para empezar, después de haber aprendido a movernos por el sistema, lo primero que veremos es cómo crear un directorio, para ello tenemos la siguiente instrucción:

mkdir [opciones] directorio [directorio2 ... directorio 3]

Donde, como podéis apreciar, podemos crear de uno a varios directorios a la vez, así que ahora toca probarlo.

```
[matados2k@fortaleza matados2k]$ mkdir curso
[matados2k@fortaleza matados2k]$ cd curso
[matados2k@fortaleza curso]$ mkdir dir1 dir2
[matados2k@fortaleza curso]$ ls --color -l
total 8
drwxrwxr-x    2 matados2k matados2k     4096 may 18 16:15 dir1
drwxrwxr-x    2 matados2k matados2k     4096 may 18 16:15 dir2
```

Como podéis ver el uso es bien sencillo, como opciones vamos a destacar '-p' que crea los directorios padres que no existan, como vemos en el siguiente ejemplo.

```
[matados2k@fortaleza curso]$ mkdir -p ./dir1/NOEXISTO/ops
[matados2k@fortaleza curso]$ cd dir1
[matados2k@fortaleza dir1]$ ls
NOEXISTO
[matados2k@fortaleza dir1]$ cd NOEXISTO/
[matados2k@fortaleza NOEXISTO]$ ls
ops
[matados2k@fortaleza NOEXISTO]$
```

Y por último la opción '-m', que nos da la posibilidad de indicarle los permisos que tendrán nuestros directorios, pero... como aún no lo hemos visto en el curso queda pendiente de que lo probéis vosotros mismos cuando veamos el tema de los permisos. La opción -m va seguido del modo antes de poner los directorios.

No me gustan los directorios

Ya que hemos aprendido a crearlos hay que aprender también la opción contraria, y ya que mkdir venía de make directory ahora necesitamos un remove directory:

rmdir [opciones] directorio [directorio2 ... directorio 3]

Así que lo que nos queda es comenzar a probarlo:

```
[matados2k@fortaleza NOEXISTO]$ rmdir ops
[matados2k@fortaleza NOEXISTO]$ cd ..
[matados2k@fortaleza dir1]$ cd ..
[matados2k@fortaleza curso]$ rmdir ./dir1/NOEXISTO/
[matados2k@fortaleza curso]$ ls
dir1  dir2
[matados2k@fortaleza curso]$ rmdir dir1 dir2
[matados2k@fortaleza curso]$ ls
[matados2k@fortaleza curso]$
```

Como podéis ver se pueden borrar varios a la vez, igual que con mkdir creábamos varios a la vez. En este caso, para rmdir solo comentaré una opción interesante y es '-p'. Esta opción lo que hace es borrar recursivamente la ruta completa que le indiquemos, por ejemplo, si tenemos /a/b/c un 'rmdir -p /a/b/c' es equivalente a 'rmdir /a/b/c' + 'rmdir /a/b' + rmdir 'a'. Vayamos al ejemplo:

```
[matados2k@fortaleza curso]$ mkdir -p ./dir1/uno/dos
[matados2k@fortaleza curso]$ ls
dir1
[matados2k@fortaleza curso]$ ls dir1
uno
[matados2k@fortaleza curso]$ ls ./dir1/uno
dos
[matados2k@fortaleza curso]$ rmdir -p dir1/uno/dos
[matados2k@fortaleza curso]$
```

Tampoco me gusta teclear tanto

Esto es algo que quizás debí contar enla entrega anterior. El intérprete de comandos, para facilitarnos nuestro trabajo, rellena por nosotros con la opción más lógica para los comandos y los archivos, simplemente empezando a escribirlos y pulsando tabulador. Por ejemplo, no nos acordamos de cómo se escribía el comando 'mkdir' y solo nos acordamos de 'mkd' pulsamos el tabulador y nos lo rellenara automáticamente o nos indicará qué comandos empiezan por 'mdk'. En mi caso tabulo y me pita, no me sale nada, vuelvo a tabular y ahora sí me da los comandos que empiezan por mdk:

```
[matados2k@fortaleza matados2k]$ mkd  (Una tabulación y nada, dos y tachan ...)
mkdep      mkdir      mkdirhier
[matados2k@fortaleza matados2k]$ mkd
```

Ahora si añado una 'i' me completa con 'mkdir', ya que 'mkdir' está contenido dentro del nombre de 'mkdirhier'. Ahora tenemos lo que buscábamos, pero si volvemos a tabular nos escribirá 'mkdirhier'.

Con lo que llegamos a la conclusión de que rellenará cuando: a) No hay otro comando que coincida con la cadena que hayamos escrito, y b) Cuando el nombre de un comando esté contenido en otro pudiendo seguir tabulando para llegar al otro.

Cuando existen muchas posibilidades, tantas que no caben en pantalla, antes nos preguntará si mostrarlo o no. Por ejemplo, ejecutar:

```
[matados2k@fortaleza matados2k]$ m
Display all 190 possibilities? (y or n)
[matados2k@fortaleza matados2k]$ m
```

En mi caso he contestado con 'n' ya que no quiero ver la lista completa. Por ejemplo, si quisiéramos ver todos los ejecutables accesibles no tendríamos más que tabular dos veces sin escribir nada.

```
[matados2k@fortaleza matados2k]$
Display all 2809 possibilities? (y or n)
[matados2k@fortaleza matados2k]$
```

Todo esto es completamente aplicable a los ficheros que indicamos a nuestros comandos, probad a escribir 'cd /e' y tabular.

Copiando voy y copiando vengo

Y una vez visto esto llegamos al momento en el que vamos a crear copias de los archivos, para ello tenemos el siguiente comando:

cp [Opciones] Origen Destino

El uso es bien sencillo, sólo hay que indicar el origen y el destino de lo que queremos copiar:

```
[matados2k@fortaleza curso]$ cp /etc/termcap .
[matados2k@fortaleza curso]$
```

Como vemos no se nos pide confirmación para copiar, y pudiera ocurrir que nos hubiésemos equivocado y sobreescribiéramos un archivo que ya tuviera sin que nos pregunte, ya que 'cp' también renombra incluyendo el nombre en el destino. Veamos un ejemplo:

```
[matados2k@fortaleza curso]$ cp /etc/shells ./termcap
[matados2k@fortaleza curso]$ ls
dir1  termcap
[matados2k@fortaleza curso]$
```

Como vemos hemos copiado el archivo 'shells' que está en '/etc' a nuestro directorio actual y con el nombre 'termcap', con lo cual hemos sobreescrito nuestro fichero original y eso puede ser algo que no deseemos. Para hacer que nos pregunte antes de copiar usamos la opción '-i' como vemos en el ejemplo:

```
[matados2k@fortaleza curso]$ cp -i /etc/termcap .
cp: ¿sobreescribir `./termcap'? (s/n) s
[matados2k@fortaleza curso]$
```

Bueno, ya hemos recuperado nuestro fichero original, menos mal ;) , pero 'cp' es aún más versátil, ya que con la opción '-r' podemos copiar directorios enteros aunque es preferible usar la opción '-R' porque '-r' no tiene un comportamiento definido (o sea que puede pasar cualquier cosa) si se copian ficheros que no son normales como pueden ser un dispositivo. Veamos un ejemplo:

```
[matados2k@fortaleza curso]$ mkdir copia_de_bin
[matados2k@fortaleza curso]$ cp -r /bin ./copia_de_bin
[matados2k@fortaleza curso]$ cd copia_de_bin/
[matados2k@fortaleza copia_de_bin]$ ls
bin
[matados2k@fortaleza copia_de_bin]$ cd bin
[matados2k@fortaleza bin]$ ls
arch           dmesg          ipcalc         ping       tcsh
.......... (omito archivos para que no ocupe tanto)...........
df             igawk          pgawk          tar
[matados2k@fortaleza bin]$
```

Y para rematar sólo queda comentar que 'cp' también tiene la opción '-p', que es igual que en 'mkdir' y 'rmdir' así que sobran las explicaciones. Por cierto, 'cp' admite más de un origen, así que puedes copiar varios archivos en una sola línea a un mismo destino.

Y borrando por el camino yo me entretengo

Si 'cp' viene de copy entonces algunos ya habréis adivinado qué orden es la de borrar:

rm [opciones] lista_de_ficheros_a_borrar

Veamos el uso más simple de rm:

```
[matados2k@fortaleza curso]$ cp termcap teborrare
[matados2k@fortaleza curso]$ ls
copia_de_bin  dir1  teborrare  termcap
[matados2k@fortaleza curso]$ rm teborrare
[matados2k@fortaleza curso]$ ls
copia_de_bin  dir1  termcap
[matados2k@fortaleza curso]$
```

Igual que 'cp' , 'rm' también tiene la opción '-i' y también puede borrar directorios enteros con '-r' y '-R' (en este caso ambos son iguales), aunque también se borran con '-d', pero '-d' no tiene un carácter recursivo y deja desenlazados los ficheros que contiene (o sea que es una cagada, así que mejor no usarla) lo que quiere decir que se quedan como diríamos sin ningún directorio que los contenga, por

lo que hay que tener mucho cuidado con esta opción.

```
[matados2k@fortaleza curso]$ rm -ri copia_de_bin dir1
rm: ¿descender al directorio `copia_de_bin'? (s/n) s
rm: ¿descender al directorio `copia_de_bin/bin'? (s/n) s
..........
rm: ¿borrar el enlace simbólico `copia_de_bin/bin/csh'? (s/n) s
rm: ¿borrar el fichero regular `copia_de_bin/bin/tcsh'? (s/n) s
rm: ¿borrar el directorio `copia_de_bin/bin'? (s/n) s
rm: ¿borrar el directorio `copia_de_bin'? (s/n) s
rm: ¿descender al directorio `dir1'? (s/n) s
rm: ¿descender al directorio `dir1/uno'? (s/n) s
rm: ¿borrar el directorio `dir1/uno/dos'? (s/n) s
rm: ¿borrar el directorio `dir1/uno'? (s/n) s
rm: ¿borrar el directorio `dir1'? (s/n) s
[matados2k@fortaleza curso]$
```

Solo queda apuntar que, tanto 'cp' como 'rm', si '-i' hace que pregunte la opción contraria es '-f' que no preguntará nada de nada.

Estamos en movimiento

Venga, que seguro que algunos ya se han imaginado que para mover es:

mv [Opciones] origen destino

Bueno, 'mv' equivale a copiar y borrar, y al igual que 'cp' admite varios orígenes y un directorio destino. Así que con los ejemplos vistos antes con 'cp' sobran los ejemplos, salvo para el caso en que el origen y el destino son el mismo, pero en el destino se indica un nombre de fichero con lo cual lo que hacemos es renombrar:

```
[matados2k@fortaleza curso]$ ls
termcap
[matados2k@fortaleza curso]$ mv termcap perro
[matados2k@fortaleza curso]$ ls
perro
[matados2k@fortaleza curso]$
```

Como veis el directorio origen y destino es el mismo, por lo cual en el destino hemos indicado un cambio de nombre.

Como 'mv' no tiene opción que digamos interesantes salvo quizás '-i' y '-f' (que es igual que en 'cp' y 'rm') pues damos por finalizada esta entrega, que por cierto se ha hecho muy larga. En la próxima entrega aprenderemos a usar los caracteres comodín, cómo consultar la ayuda de Linux y el historial de órdenes.

Última hora.

Si te gusta la propuesta que lanzo a todos y quieres participar, envíame la extensión que creas conveniente a cualquiera de los e-mail que indico en los documentos, ocomenta que quieres participar.

Entrega 5. Usando comodines, manuales e históricos.

```
$ \(-
bash: (-: command not found
```

Un asterisco para dominarlos a todos.

Comenzamos de nuevo otra entrega más. Vamos a ver el uso de los comodines, que no es ni más ni menos que una característica del intérprete de comandos que nos permite referirnos a un conjunto de ficheros a la vez.

Empezaremos viendo primero el comodín '*': el comodín '*' hace referencia a cualquier carácter o cadena de caracteres (es decir, sustituye a uno, ninguno o muchos caracteres). Para entenderlo bien, ya que la explicación puede ser un poco confusa, veamos como siempre un ejemplo sencillo:

```
[matados2k@fortaleza matados2k]$ cd /dev
[matados2k@fortaleza dev]$ ls l*
lirc    logimouse  loop10  loop13  loop2  loop5  loop8  lp1
log     loop0      loop11  loop14  loop3  loop6  loop9  lp2
logibm  loop1      loop12  loop15  loop4  loop7  lp0    lp3

logicalco:
bci  dci1300
[matados2k@fortaleza dev]$
```

Lo que acabamos de indicarle a ls con el uso de '*' es que nos liste todos los ficheros que empiecen por 'l' seguido de cualquier cosa, incluso nos lista el contenido de un directorio que empieza por 'l'. Otro ejemplo, para que nos quede definitivamente claro, puede ser este:

```
[matados2k@fortaleza dev]$ ls *rr*
stderr
[matados2k@fortaleza dev]$
```

En este caso lo que hemos hecho es decirle que nos liste todos los ficheros que contengan la cadena 'rr' (incluso los que empiecen o terminen por 'rr' ya que '*' incluso puede ser ningún carácter).

Si sois observadores y/o curiosos veréis que no funciona con los ficheros ocultos (acordaos de que empiezan por '.'), porque afectaría entre otras cosas a los directorios especiales '.' y '..' que ya expliqué en entregas anteriores, y según con qué comandos lo utilicemos puede provocar verdaderos problemas (como por ejemplo rm).

Otra cosa importante es que en los ejemplos anteriores no es el comando 'ls' el que recibe por ejemplo el argumento 'l*', sino que es el intérprete de comandos (en mi caso bash) el que se encarga de buscar las coincidencias y lo que haría es ejecutar algo como 'ls lirc logimouse loop10 loop13 loop2 ....' . Lo que quiero decir es que no es el comando el que procesa los comodines, sino el propio intérprete de

comandos, que se encarga luego de llamar al comando con los comodines ya procesados.

Y una interrogación para cada señor de la tierra GNU.

El siguiente y último comodín es '?' que hace referencia a cualquier carácter, pero en este caso sólo a uno (no puede ser ninguno como el caso de '*'). Para verlo claro veamos un ejemplo (para variar):

```
[matados2k@fortaleza dev]$ ls ?l??
tlk0  tlk1  tlk2  tlk3
[matados2k@fortaleza dev]$
```

En este caso le preguntamos a ls por todos aquellos comandos que tienen longitud 4 y el segundo carácter es una 'l', ¿sencillo, verdad?

Puedes combinar tanto '*' como '?' para conseguir el resultado deseado:

```
[matados2k@fortaleza dev]$ ls ?l*1
aloadC1  tlk1
[matados2k@fortaleza dev]$
```

Lo que hemos hecho es decirle a 'ls que nos muestre todos aquellos que de segundo carácter tienen una 'l' seguido de cualquier número de caracteres y acabe en 1. Quizás con 'ls' no le veáis mucha utilidad a estos comodines, pero imaginaos con cp, rm y mv por ejemplo. Podéis hacer tantas combinaciones como se os ocurran, así que a probad vosotros mismos.

Consultando información.

Muchas veces necesitamos conocer más acerca de un comando determinado o del uso del propio intérprete y nos gustaría tener un manual, en el caso de GNU/linux disponemos de un manual en línea muy útil denominado 'man'. Su sintaxis básica es la siguiente:

man [sección] comando

Por ejemplo, para saber todo de 'cp' no hay mas que consultar 'man cp' nos moveremos con los cursores arriba o abajo y para salir pulsamos la letra 'q'. La barra espaciadora pasa página a página y podemos usar Re Pag y Av Pag de nuestro teclado para movernos.

El manual en línea está dividido en secciones, concretamente de la 1 a la 9, cada una referidas a una cosa distinta. Por ejemplo, la sección 2 es la de programación de C. Por defecto no pondremos sección y encontraremos lo que buscamos ya que busca en todas, pero en ocasiones (como por ejemplo cuando se programa) hay funciones que se llaman igual que algún comando de Linux y por defecto nos dará el comando en vez de la función. En ese caso, por ejemplo, le especificamos que queremos la sección 2. Para ver por vosotros mismos cómo se usa más a fondo 'man' probad con 'man man'.

Otra fuente útil de ayuda en línea es sin duda el comando 'info', que tiene un manejo para visualizar la información igual que 'man'.

Siguiendo tus propios pasos.

Para no tener que escribir una y otra vez los mismos comandos, el intérprete de comando mantiene un histórico de las órdenes que introducimos. Para mostrarlas usad las teclas decursor arriba y abajo.

Para ver todo el historial de golpe tenemos el comando 'history', que nos mostrará numeradas todas y cada una de las órdenes que le hemos dado al intérprete de comandos. Esto es especialmente útil para ejecutar directamente un comando del historial usando '!' seguido del número de orden visto con 'history'. Por ejemplo, para la orden 5 del historial ejecutad '!5'.

En el caso de bash el historial es guardado en nuestro propio directorio de trabajo, en el fichero '.bash_history', en el cual podremos visualizar y manipular su contenido a nuestro gusto. (Recordad que es un fichero oculto).

Despedida.

Siento que esta entrega sea en esta ocasión bastante corta, pero es que el tema no da más de sí y mi tiempo tampoco. En la próxima entrega empezaremos con el uso de la fontanería Unix, como es el uso de la Entrada/Salida estándar, redireccionamiento, pipes y demás.

Entrega 6. Viva la fontanería.

% !bluemoon
bluemoon: Event not found.

Entrada y salida estándar.

Normalmente los comandos usan lo que esconocido como entrada y salida estándar (stdin y stdout respectivamente, lo puedes encontrar en /dev/stdin y /dev/stdout) que no son ni más ni menos que el teclado y la pantalla.

Para ver esto usaremos el siguiente comando:

cat [opciones] [lista_de_ficheros]

'Cat' en sí es un concatenador de ficheros que imprime la salida por la salida estándar, las opciones no las veremos puesto que no son muy interesantes. Nos va a servir para visualizar los ficheros y si escribimos 'cat' sin parámetros leerá de la entrada estándar y escribirá en la salida estándar.

```
[matados2k@fortaleza curso]$ cat
hola
hola
a
a
todos
todos
[matados2k@fortaleza curso]$
```

Vemos que sale repetido todo como si 'cat' nos vacilara, simplemente repetirá todo lo que escribamos por que es lo que realmente sabe hacer. Para salir de 'cat' debemos mandarle la señal EOT (End-of-text, fin de texto) y para eso tenemos que usar CTRL+D.

Algunos pensaréis "qué estupidez de comando", pero sólo hasta que sepáis que con 'cat' se hacen cosas tan curiosas como una imagen ISO de CD sin usar ninguna opción, pero eso será mucho más adelante.

Redirección de salida y entrada.

El intérprete de comandos nos permite redireccionar la salida estándar a un fichero usando el símbolo '>'. Para ver cómo funciona qué mejor que un ejemplo:

```
[matados2k@fortaleza curso]$ cat > hola_holita
hola
holita
vecinitos
soy ned
[matados2k@fortaleza curso]$ cat hola_holita
hola
holita
vecinitos
soy ned
[matados2k@fortaleza curso]$
```

Podemos ver que usamos 'cat' con una redirección a un fichero llamado 'hola_holita', podemos ver que no nos ha repetido lo que escribimos puesto que la salida está redireccionada y después visualizamos su contenido con el mismo 'cat'.

La redirección de entrada es similar, sólo que se usa el carácter '<'. Veamos un ejemplo:

```
[matados2k@fortaleza curso]$ cat < hola_holita
hola
holita
vecinitos
soy ned
[matados2k@fortaleza curso]$
```

En este ejemplo vemos que el resultado es el mismo y es obvio, ya que esta vez hemos cambiado la entrada estándar por un fichero y la ha sacado por la salida estándar, con lo que ha repetido lo que había en el fichero como si lo tecleásemos y ha terminado ya que los ficheros contienen el carácter EOT o EOF (End of File).

Hay que destacar que la redirección de salida es una redirección destructiva, con esto quiero decir que si no existe el fichero lo crea y si existe lo sobrescribe, y esto puede ser algo que no deseemos.

Redirección de salida no destructiva.

Imaginemos que vamos a hacer una lista de la compra para un piso que tenemos de estudiantes, entonces escribiríamos algo así:

```
[matados2k@fortaleza curso]$ cat > lista_compra
Champu
Gominolas
Cerveza
Panchitos
jb
[matados2k@fortaleza curso]$
```

Y se nos olvidan las aspirinas:

```
[matados2k@fortaleza curso]$ cat > lista_compra
aspirinas
[matados2k@fortaleza curso]$ ls
hola_holita  lista_compra  perro
[matados2k@fortaleza curso]$ cat lista_compra
aspirinas
[matados2k@fortaleza curso]$
```

Pues nada ya estropeamos la fiesta, ya que la lista se la dimos a Manolito y sólo nos trajo aspirinas. Para evitar algo tan absurdo y surrealista como esto tenemos la redirección de salida no destructiva, para ello usamos '>>', veamos el ejemplo:

```
[matados2k@fortaleza curso]$ cat >> lista_compra
Manolito no te olvides de:
Champu
Gominolas
Cerveza
Panchitos
jb
[matados2k@fortaleza curso]$ cat lista_compra
aspirinas
Manolito no te olvides de:
Champu
Gominolas
Cerveza
Panchitos
jb
[matados2k@fortaleza curso]$
```

Con esto ya nos ahorramos el problema. Supongo que ya habréis deducido ventajas de la redirección aparte de hacer la lista de la compra, ¿no? (tan simples como guardar el listado de 'ls' en un fichero y tan complicado como ahorrarnos teclear en comandos que nos piden muchos datos).

Usando tuberías. Las pipes.

Para lo siguiente vamos a ver otro comando que lo que hace es ordenar lo que le entra y devolverlo ordenado.

sort [opciones] [lista_de_ficheros]

No vamos a ver las opciones por el momento (siempre puedes hacer 'man sort' si te interesa). Veamos antes de todas formas cómo se comporta sort sin opciones:

```
[matados2k@fortaleza curso]$ sort
peras
limones
piñas
melocotones
(aquí pulsé CTRL+D)
limones
melocotones
peras
piñas
[matados2k@fortaleza curso]$
```

Observamos que espera a que pulsemos EOT y nos devuelve todo ordenado.

Seria muy interesante poder unir la salida de un programa con la entrada de otro y construir una cadena de órdenes. Imaginaos que en el caso anterior si a Manolito no le ordenamos la lista se pierde (no se molesten los Manolos, pero algún nombre tendría que tener la víctima), lo mejor sería hacer la lista con 'cat', unir la salida con 'sort' y redireccionar al fichero donde queremos guardarlo, y para eso usamos el carácter '|' (la del AltGr +1). Lo que hacemos con ese símbolo es crear una pipe, que es como ya expliqué unir la salida de un comando con la entrada de otro. Mejor un ejemplo:

```
[matados2k@fortaleza curso]$ cat < lista_compra | sort > nueva_lista
[matados2k@fortaleza curso]$ cat nueva_lista
aspirinas
Cerveza
Champu
Gominolas
jb
Manolito no te olvides de:
Panchitos
[matados2k@fortaleza curso]$
```

Observamos que redireccionamos la entrada de 'cat' con la lista ya creada y construimos una pipe para unir la salida de 'cat' con la entrada de 'sort' y terminamos redireccionando la salida a un nuevo fichero. Con lo que conseguimos la lista ordenada, que si bien podíamos haber hecho lo mismo con

'sort < lista_compra > nueva_lista' tenía que hacerlo de la otra forma para explicarlo.

Despedida.

Ya con esto terminamos, y para la próxima entrega nos tocan los permisos de los ficheros y la creación de enlaces.

Entrega 7. ¡Permiso!

% cd ~god
Unknown user: god.

Los permisos de ficheros.

Como Linux es un sistema multiusuario, debemos proteger los ficheros dela manipulación por parte de otros. Linux nos proporciona para esto los conocidos permisos de ficheros. Sin más acordaos de 'ls -l':

```
[matados2k@fortaleza curso]$ ls -l
total 740
-rw-rw-r--    1 matados2k matados2k       30 jun  4 16:07 hola_holita
-rw-rw-r--    1 matados2k matados2k       75 jun  4 16:27 lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$
```

En la entrega 3 ya quedamos claro que la primera parte (-rw-rw-r--) no la explicaría hasta que no llegara a esta entrega.

Los permisos de ficheros los podemos establecer en 3 niveles, permisos del propietario (usuario o user), permisos para el grupo (group) y permiso para el resto (others).

Cada fichero es del usuario que lo crea o bien los obtiene porque alguien le ha cambiado la propiedad (por ejemplo, root puede hacer esto). Sólo el propietario del fichero y el superusuario (root) pueden cambiar los permisos de los ficheros.

Cada usuario puede pertenecera uno o a varios grupos de usuarios, y es a los usuarios que están dentro del mismo grupo que el propietario del fichero a quienes afectan los permisos de grupo.Y evidentemente los permisos para el resto afectan a todos los demás.

Y a la vez de todo esto hay 3 tipos de permisos: permisos de lectura, permisos de escritura y permiso de ejecución.

Interpretando los permisos.

Bien, como ya hemos visto con la opción '-l' de 'ls' podemos observar los permisos que tiene cada fichero asignado, siendo una cadena de 10 caracteres (por ejemplo -rw-rw-r--). El primer carácter no lo explicaremos puesto que ya lo hicimos en la entrega 3.

Con esto ya nos quedan 9, que lo dividiremos en partes de 3: el primero para la lectura, el segundo para la escritura y el tercero para la ejecución (si sois avispados ya os habréis dado cuenta de que en Linux no se ejecutan los ficheros por tener un tipo de extensión, si no por tener o no este permiso). Y el orden es el mismo: primero los 3 de usuario, los 3 de grupo y los 3 de los otros (buena película ;)). Un '-' indica que ese permiso no está dado.

Para que quede claro veamos unos ejemplos:

```
-rw-rw-r-- El usuario puede leer y escribir, el grupo puede leer y escribir y el
resto solo leer.

---------- Nadie puede hacer nada.
-rwxrwxrwx Todos pueden hacer todo.
-rwx------ El usuario puede hacer todo.
---x--x--x El fichero solo puede ejecutarse por todos.
-rwxr----- El usuario puede hacerlo todo y el grupo solo leer
```

Depende... ¿de qué depende?

Hay que indicar que los permisos de los ficheros dependen de los permisos en sí del directorio que los contiene, y que de nada sirve tener '-rwxrwxrwx' en un fichero si el directorio sólo tiene permisos '-r--------', con lo cual sólo podríamos leerlo y nada más. Incluso si el directorio no tuviera permiso de lectura para nadie, no podríamos ni siquiera listar el contenido del directorio, para ver esto con un usuario normal intentad haced lo siguiente:

```
[matados2k@fortaleza curso]$ ls /root
ls: /root: Permiso denegado
[matados2k@fortaleza curso]$
```

Según la configuración de vuestro sistema puede que os deje, si es así seguramente no tendréis una buena seguridad en vuestro sistema.

Como curiosidad a esto, por ejemplo, podéis crear un buzón donde todos pueden entrar los ficheros que quieran pero sólo tu puedes verlo y manipularlo, sería algo así como '-rwx-w--w-'.

Cambiando permisos.

De momento sólo veremos cómo cambiar los permisos a los ficheros, este comando es:

chmod [opciones] modo fichero

Como opción únicamente comentaremos '-R' que hace que el cambio de permisos sea recursivo, por ejemplo, para dar los permisos a todos los ficheros de un directorio y sus descendientes.

Para el modo de momento sólo vamos a ver la forma amigable de hacerlo, ya que se puede hacer con un número en octal. Para esto describiremos el modo como {a,u,g,o}{+,-}{rwx} donde:

```
a  Indica que se aplicará a todos.(all)
u  Indica que se aplicará al usuario.(user)
g  Indica que se aplicará al grupo.(group)
o  Indica que se aplicará a otros.(other)
+  Indica que se añade el permiso.
-  Indica que se quita el permiso.
r  Indica permiso de lectura.
w  Indica permiso de escritura.
x  Indica permiso de ejecución.
```

Así que primero indicamos a quien vamos aplicar el permiso, y después qué permiso pondremos o quitaremos. Vemos ejemplos del modo:

```
a+r   Permisos de lectura para todos.
+r    Igual que antes, si no se indica nada se supone 'a'.
og-x  Quita permiso de ejecución a todos menos al usuario.
u+rwx Da todos los permisos al usuario.
o-rwx Quita los permisos a los otros.
```

Con todo esto sólo queda hacer un ejemplo de la ejecución de 'chmod':

```
[matados2k@fortaleza curso]$ ls -l
total 740
-rw-rw-r--    1 matados2k matados2k       30 jun  4 16:07 hola_holita
-rw-rw-r--    1 matados2k matados2k       75 jun  4 16:27 lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$ chmod a+rwx hola_holita
[matados2k@fortaleza curso]$ ls -l
total 740
-rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
-rw-rw-r--    1 matados2k matados2k       75 jun  4 16:27 lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$
```

Sólo comentar que con 'chmod' podemos hacer aún más, pero eso lo veremos cuando veamos la gestión de usuarios.

Entrega 8. Enlaces y tareas

```
%touch me
touch: cannot touch me: permission denied
```

¿Para qué sirven los enlaces?

Los enlaces sirven para dar a un fichero múltiples nombres (no confundáis los que venís del mundo de Windows con accesos directos). Internamente para el sistema los ficheros son identificados por un número que se llama **inodo**, lo que quiere decir que el nombre de un fichero está asociado a un inodo que es lo que usa realmente el sistema operativo, el cual es el único identificador para el sistema. Con esto podríamos decir que simplemente los nombres esos que vemos al hacer un 'ls' son **enlaces** a inodos. Y como podréis deducir un directorio no es más que una lista de inodos con sus respectivos nombres de ficheros.

Enlaces duros.

Un enlace duro consiste en asignar un nombre de fichero a un inodo, con lo cual podemos referenciar a un mismo fichero con varios nombres y un cambio en uno de ellos implica un cambio en el resto ya que se trata del mismo fichero realmente. Para esto usamos el comando:

ln [Opciones] Origen Destino

Donde el origen es el nombre del fichero del cual queremos hacer un enlace duro, y para ver el número de inodo usaremos la opción '-i' de 'ls'. Para que todo quede claro un ejemplo:

```
[matados2k@fortaleza curso]$ ls -i
1264303 hola_holita  1264304 lista_compra  1264305 nueva_lista  1264193 perro
[matados2k@fortaleza curso]$ ln lista_compra lista2
[matados2k@fortaleza curso]$ ls -i lista_compra lista2
1264304 lista2  1264304 lista_compra
[matados2k@fortaleza curso]$
```

Lo primero que hacemos es mirar todos los inodos de los ficheros contenidos en nuestro directorio, creamos un enlace a lista_compra llamado lista2 y comprobamos que efectivamente tienen el mismo inodo.

Lo mismo ya os lo estáis preguntando: ¿qué pasa si borro uno de ellos? Si os acordáis de la entrega 3 hablábamos que una de las cosas que nos mostraba 'ls -l' es el número de enlaces de un fichero, pues si borras uno de ellos simplemente el contador disminuye en 1, y cuando no hay más enlaces es borrado realmente. Veámoslo mejor con un ejemplo:

```
[matados2k@fortaleza curso]$ ls -l
total 744
-rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
-rw-rw-r--    2 matados2k matados2k       75 jun  4 16:27 lista2
-rw-rw-r--    2 matados2k matados2k       75 jun  4 16:27 lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$
[matados2k@fortaleza curso]$ rm lista2
[matados2k@fortaleza curso]$ ls -l
total 740
-rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
-rw-rw-r--    1 matados2k matados2k       75 jun  4 16:27 lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$
```

Podemos observar con el primer 'ls -l' que lista2 y lista_compra tienen ambos un 2 ya que al inodo al que apuntan tiene 2 enlaces, y como ambos apuntan al mismo pues ambos muestran que tienen 2. Y al borrar uno de ellos simplemente se elimina el enlace y disminuye el contador. Fácil, ¿verdad?

Hay que resaltar que los enlaces duros tienen una limitación y es que sólo se pueden hacer enlaces de este tipo en un mismo sistema de ficheros. Con esto quiero decir que si tienes por ejemplo dos particiones no puedes hacer enlaces duros entre particiones. Por ejemplo, yo tengo varias particiones para distintas partes del árbol de ficheros como son '/', '/home' y '/usr' todas ellas de tipo Ext3. Pues bien aquí un ejemplo de un enlace duro entre particiones:

```
[matados2k@fortaleza curso]$ ln lista_compra /etc/lista_chula
ln: creando el enlace duro `/etc/lista_chula' a `lista_compra': Enlace cruzado
entre dispositivos no permitido
[matados2k@fortaleza curso]$
```

Otros inconvenientes son que no se puede hacer un enlace a un fichero que no existe, y es difícil saber a qué fichero apunta un enlace. Pero para esto no se vayan todavía porque aún hay más ...

Enlaces simbólicos.

Éstos son otros tipo de enlaces bastante diferentes y sin las limitaciones anteriormente dichas. Este tipo de enlaces permite dar a un fichero el nombre de otro, pero no enlaza el fichero con un inodo. Lo que hace en este caso es realmente apuntar al nombre del fichero con quien enlaza, por lo cual tendrán inodos diferentes y si uno se elimina no se elimina el otro (esto es ya más parecido a lo que es un acceso directo de Windows, aunque no lo es para nada).

Para realizar este tipo de enlaces usamos la opción '-s' de 'ln' y para entenderlo mejor como siempre un ejemplo:

```
[matados2k@fortaleza curso]$ ls -li
total 740
1264303 -rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
1264304 -rw-rw-r--    1 matados2k matados2k       75 jun  4 16:27 lista_compra
1264305 -rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
1264193 -rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$ ln -s lista_compra lista2
[matados2k@fortaleza curso]$ ls -li
total 740
1264303 -rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
1264331 lrwxrwxrwx    1 matados2k matados2k       12 jun 22 16:56 lista2 ->
lista_compra
1264304 -rw-rw-r--    1 matados2k matados2k       75 jun  4 16:27 lista_compra
1264305 -rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
1264193 -rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$
```

Podemos ver en este ejemplo cómo los inodos son totalmente distintos, y cómo lista2 es un enlace a lista_compra de una manera muy sencilla. Y para ver que lo anteriormente dicho es cierto eliminaremos lista_compra y quedará lista2.

```
[matados2k@fortaleza curso]$ rm lista_compra
[matados2k@fortaleza curso]$ ls -l
total 736
-rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
lrwxrwxrwx    1 matados2k matados2k       12 jun 22 16:56 lista2 -> lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$
```

¿Y adivinan ya qué pasa si hacemos 'cat' a lista2, o hace falta un ejemplo? ;)

Control de tareas.

El control de tareas es una utilidad incluida en la mayoría de los intérpretes de comandos, y permiten el control del multitud de tareas o comandos a la vez en un solo terminal.

Los procesos pueden estar en Primer Plano o en Segundo Plano. En primer plano solamente puede haber un proceso a la vez y ese es precisamente el que vemos, en otras palabras el que interactúa con nosotros dándonos una salida o pidiendo datos, el que recibe las órdenes del teclado. En cambio el proceso en segundo plano no recibe señal (normalmente) desde el teclado y se ejecuta en silencio.

Si un programa tarda mucho en terminar y no muestra nada interesante por pantalla lo más lógico sería ejecutarlo en segundo plano, como pudiera ser la compresión de un archivo enorme. Si lo hacemos así podremos seguir usando el ordenador para hacer cualquier otra cosa.

Un proceso puede ser suspendido (dormido), lo que indica que temporalmente estará parado y más tarde podríamos reanudarlo para cuando lo necesitemos.

O bien cuando ya no nos interese un proceso podremos interrumpirlo y acabar con él.

Despedida.

Ya que se acaba el espacio de esta semana, la próxima continuaremos con esto viendo cómo pasamos tareas de primer a segundo plano, eliminación y suspensión de proceso y su puesta en marcha de nuevo. Así que no olviden "Mineralizarse y Vitaminarse" como ya decía super ratón en nuestra infancia ;) Hasta la próxima.

Entrega 9. Continuamos con el control de tareas.

```
%touch me
touch: cannot touch me: permission denied
```

Mata que te mata.

Después de la introducción en la anterior entrega vamos a ponernos manos a la obra. Para ello vamos a ver un comando que para mi es casi totalmente inútil pero que nos servirá muy bien para las explicaciones, es el comando:

yes [string]

Este comando simplemente va a repetir lo que le indiquemos en string indefinidamente y en su defecto nos dará una ristra de 'y' si no le indicamos nada. En los manuales de los que yo aprendí ponía que era utilísimo para no tener que estar diciendo siempre que si a un comando (recordar el uso de redirección y pipes), pero, ¿realmente alguien le ve la utilidad a decir todo que si sin leer antes?.

Paro probarlo tecleemos yes y haber que pasa:

```
[matados2k@fortaleza curso]$ yes
y
y
y
....
```

Ya aburridos de ver 'y' sin parar lo que vamos hacer el matarlo por cansino y aburrido ;) paraello pulsamos 'Ctrl + c' y se acabo.

Como al hacer la pruebas no nos interesa estar mirando un montón de 'y' lo que haremos es redirigir la salida del comando 'yes' hacia '/dev/null' que es algo así como un agujero negro, es decir todo lo que allí mandamos desaparece (mas de uno querría un '/dev/null' en la vida real).

Enviando al segundo plano.

Ahora lo que queremos hacer es arrancar el comando yes con la redirección a '/dev/null' y arrancarlo en segundo plano consiguiendo un bonito gasto de cpu :). Para ello debemos añadir el simbolo '&' al final de la línea, como vemos en el ejemplo.

```
[matados2k@fortaleza curso]$ yes > /dev/null &
[1] 5064
[matados2k@fortaleza curso]$
```

Este es el resultado, volvemos a tener el prompt para nosotros y dos curiosos números. Esos dos números son las referencias a la tarea, el '[1]' representa el número de tarea y el 5064 es el identificador de proceso o PID que es ni mas ni menos el número que le asigna el sistema al proceso.

Pero ¿donde se ha metido? .

¿Y ahora que hacemos para ver que a pasado con nuestro querido comando? Pues simple, usamos unos de los comandos internos del interprete de comando, este es 'jobs' que da un listado de las tareas que tenemos en ejecución y su estado:

```
[matados2k@fortaleza curso]$ jobs
[1]+  Running                 yes >/dev/null &
[matados2k@fortaleza curso]$
```

Como veis nos indica el número de tarea, el estado (en esta caso corriendo o mejor dicho ejecutándose) y la forma en que lo llamamos.

Mas formas de matar.

Siguiendo con nuestras ganas de eliminar a 'yes' de nuestro sistema (ahora no nos vale 'Ctrl + c') tenemos 2 formas de hacerlo con un mismo comando, este es 'kill' que así sin mas envía una señal de fin de proceso al proceso que le indiquemos.

Para hacerlo tenemos dos formas de referenciarlo, la mas cómoda sería con el número de tarea y para ello usamos el símbolo '%' de la siguiente manera:

```
[matados2k@fortaleza curso]$ kill %1
[matados2k@fortaleza curso]$
```

Así que recordad para usar el número de tarea hay que usar un '%' delante y para usar el PID simplemente pondremos el PID. Así que 'kill %1' y 'kill 5064' serian equivalentes.

Y para cerciorarnos de su defunción consultamos a nuestro querido 'jobs':

```
[matados2k@fortaleza curso]$ jobs
[1]+  Terminado               yes >/dev/null
[matados2k@fortaleza curso]$
```

Hay que destacar que 'jobs' guarda la información del proceso muerto para mostrarla 1 vez ya que si ejecutamos de nuevo 'jobs' ya nada se sabrá de la tarea muerta:

```
[matados2k@fortaleza curso]$ jobs
[matados2k@fortaleza curso]$
```

Parar y seguir.

Hay otra forma de mandar una tarea a segundo plano y es ejecutarla normalmente y hacerla suspenderse o dormirse, para ello usaremos la combinación de teclas ' Ctrl+z ':

```
[matados2k@fortaleza curso]$ yes > /dev/null
[1]+  Stopped                 yes >/dev/null
[matados2k@fortaleza curso]$ jobs
[1]+  Stopped                 yes >/dev/null
[matados2k@fortaleza curso]$
```

Para volverla a reanudar en primer plano usaremos el comando interno 'fg' (foreground) continuando la tarea justo donde se dejo (hay que recalcar que cuando una tarea esta parada no usa tiempo de cpu).

```
[matados2k@fortaleza curso]$fg
yes >/dev/null


[1]+  Stopped                 yes >/dev/null
[matados2k@fortaleza curso]$
```

Vemos como la hemos reanudado y la hemos vuelto a parar, pero en este caso ya si que la vamos a mandar al segundo plano con el comando interno 'bg' (background) que continuara con la ejecución del comando en segundo plano.

```
[matados2k@fortaleza curso]$ bg
[1]+ yes >/dev/null &
[matados2k@fortaleza curso]$
```

Recapitulando.

Las tareas pueden estar en primer y segundo plano y estas pueden estara bien ejecutándose, paradas o terminadas (muertas). Las tareas en segundo plano no pueden pararse ni eliminarlas mediante'Ctrl + c' si no que antes hay que pasarlas a primer plano para poder realizar esto.

Una tarea parada no consume tiempo de cpu, pero sigo manteniendo la memoria que estuviese usando, simplemente espera a que se le de la oportunidad de continuar con su tarea.

Aunque una tarea en segundo plano no puede recibir nada desde el teclado si que puede enviardatos a la pantalla y puede resultar muy muy molesto.

Por ejemplo si se te hubiera ocurrido ejecutar 'yes &' no podridas detenerlo con 'Ctrl+c' o con 'Ctrl + z' ya que esta en segundo plano, pero nos esta abrumando con una ristra interminables de 'y'que no nos deja ver nuestro prompt, con lo que para remediar esto debemos teclear a ciegas 'fg' para traernos la tarea al primer plano y esta vez si la podremos parar por ese método (también podríamos haberla eliminado con un 'kill' a ciegas con su número de tarea).

Mas sobre 'fg' y 'bg'.

Estos dos comandos actúan sobre sobre el último proceso parado o creado, indicado por un '+' cuando ejecutamos 'jobs' (mirad los ejemplos anteriores).Si tuviéramos varias tareas para indicarle a 'fg' y 'bg' sobre cual actuar utilizaríamos como con 'kill' el símbolo '%' mas su número de tarea.

```
[matados2k@fortaleza curso]$ yes > /dev/null &
[2] 5592
[matados2k@fortaleza curso]$ yes > /dev/null &
[3] 5593
[matados2k@fortaleza curso]$ yes > /dev/null &
[4] 5594
[matados2k@fortaleza curso]$ jobs
[1]   Running                 yes >/dev/null &
[2]   Running                 yes >/dev/null &
[3]-  Running                 yes >/dev/null &
[4]+  Running                 yes >/dev/null &
[matados2k@fortaleza curso]$ fg %2
yes >/dev/null

[2]+  Stopped                 yes >/dev/null
[matados2k@fortaleza curso]$ jobs
[1]   Running                 yes >/dev/null &
[2]+  Stopped                 yes >/dev/null
[3]   Running                 yes >/dev/null &
[4]-  Running                 yes >/dev/null &
[matados2k@fortaleza curso]$ kill %1
[matados2k@fortaleza curso]$ kill %4
[1]   Terminado               yes >/dev/null
[matados2k@fortaleza curso]$
```

Despedida.

Y ya visto este ejemplo mas complejo terminamos con el control de tareas. En la próxima entrega veremos una colección de comandos que nos serán de gran utilidad, asta la semana que viene ;)

Entrega 10. Colección de comandos.

```
% cat "door: paws too slippery"
cat: cannot open door: paws too slippery
```

Toqueteando el tiempo 'touch'.

touch [-acm][-r archivo_referencia |-t fecha] archivos...

El comando touch en su forma mas simple (sin opciones) actualiza los registros fecha y hora a la hora y fecha actual de la lista de ficheros que le indiquemos. Hay que tener en cuenta que si el fichero no existe nos lo creará. Hoy dos tipos de fechas de un archivo el de modificación (el que normalmente vemos) y el de último acceso, se cambian ambas. Aquí un ejemplo:

```
[matados2k@fortaleza curso]$ ls -l
total 736
-rwxrwxrwx    1 matados2k matados2k       30 jun  4 16:07 hola_holita
lrwxrwxrwx    1 matados2k matados2k       12 jun 22 16:56 lista2 -> lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 may 19 17:47 perro
[matados2k@fortaleza curso]$ touch hola_holita perro soy_nuevo
[matados2k@fortaleza curso]$ ls -l
total 736
-rwxrwxrwx    1 matados2k matados2k       30 jul  6 17:20 hola_holita
lrwxrwxrwx    1 matados2k matados2k       12 jun 22 16:56 lista2 -> lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:10 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 jul  6 17:20 perro
-rw-rw-r--    1 matados2k matados2k        0 jul  6 17:20 soy_nuevo
[matados2k@fortaleza curso]$
```

Como veis se han cambiado las fechas de los 2 primeros y se ha creado un tercero por no existir antes. Pata evitar que se creen nuevos ficheros se usa la opción '-c'.

Si queremos que solo nos modifique la fecha de modificación usamos la opción '-m' y si queremos que solo cambie la fecha de último acceso usamos la opción '-a', por lo tanto ya deducís que usar 'a' y 'c' a la vez es absurdo :P . Y os estaréis preguntando como ver el tiempo del último acceso verdad, pues ni mas ni menos que usando la opción '-u' junto con la de '-l' con el comando 'ls':

```
[matados2k@fortaleza curso]$ ls -lu
total 736
-rwxrwxrwx    1 matados2k matados2k       30 jul  6 17:20 hola_holita
lrwxrwxrwx    1 matados2k matados2k       12 jul  6 17:38 lista2 -> lista_compra
-rw-rw-r--    1 matados2k matados2k       75 jun  4 18:11 nueva_lista
-rw-r--r--    1 matados2k matados2k   740199 jul  6 17:20 perro
-rw-rw-r--    1 matados2k matados2k        0 jul  6 17:20 soy_nuevo
[matados2k@fortaleza curso]$
```

Con la opción '-r' tomamos un archivo de referencia y con la '-t' una fecha que le indiquemos y no se pueden usar las dos a la vez por eso en la sintaxis del comando aparece el símbolo '|' que indica que es una cosa o la otra.

El comando y casi todos los que vemos tienen mas opciones, como siempre explicare las mas básicas y/o útiles y si queréis saber mas ya sabéis recurrir al 'man'.

Cuanto ocupa todo 'du'.

du [opciones] [ficheros...]

Este comando contabiliza el espacio que ocupa en disco un fichero o un directorio con todos sus subdirectorios. Como opciones solo comentaremos 3 de ellas, la primera '-a' muestra además el espacio que ocupa cada uno de los ficheros que hay en los directorios que le indiquemos. La siguiente opción es '-b' que en vez de mostrarnos el tamaño en kB lo mostrará en bytes (1kB=1024bytes). Y por último la opción '-s' que informa solamente del directorio que le hayamos indicados sin contar sus subdirectorios.

```
[matados2k@fortaleza curso]$ cd ..
[matados2k@fortaleza matados2k]$ du curso
740     curso
[matados2k@fortaleza matados2k]$ du -a curso
4       curso/hola_holita
728     curso/perro
0       curso/soy_nuevo
4       curso/nueva_lista
0       curso/lista2
740     curso
[matados2k@fortaleza matados2k]$ du -b curso
744412  curso
[matados2k@fortaleza matados2k]$ du -s curso
740     curso
[matados2k@fortaleza matados2k]$ cd curso
```

Como observáis en el ejemplo tanto 'du' a secas como 'du -s' da el mismo resultado y no es ni mas ni menos por que el directorio 'curso' con contiene subdirectorios.

Seguimos interesados en saber cuanto ocupa 'df'.

df [opciones] [fichero...]

El comando 'df' resume la cantidad de espacio que se usa y la que se dispone en el sistema de ficheros. Ejemplo:

```
[matados2k@fortaleza curso]$ df
S.ficheros         Bloques de 1K   Usado    Dispon Uso% Montado en
/dev/hda11              3099260    574984   2366844  20% /
/dev/hda10               101054      8327     87510   9% /boot
/dev/hda6              20641788   6806000  13835788  33% /home
/dev/hda8              10480160   9622168    857992  92% /home/mnt/auxi1
/dev/hda9              14669032  10906040   3762992  75% /home/mnt/auxi2
/dev/hda7              15720264  15385624    334640  98% /home/mnt/juegos
none                     257332         0    257332   0% /dev/shm
/dev/hda12              7123608   3845140   2916608  57% /usr
/dev/cdrom1              668832    668832         0 100% /mnt/cdrom1
[matados2k@fortaleza curso]$
```

Como podéis observar tengo bastante cargado mi disco duro y soy un poco raro a la hora de elegir donde montar mis particiones ;) .

Si le indicamos a 'df' un fichero sólo nos indicará el espacio usado y disponible de la partición donde se encuentre ese fichero:

```
[matados2k@fortaleza curso]$ df .
S.ficheros         Bloques de 1K   Usado    Dispon Uso% Montado en
/dev/hda6              20641788   6809032  13832756  33% /home
[matados2k@fortaleza curso]$
```

Como opción muy interesante esta '-i' que nos muestra el uso de inodos en vez del espacio:

```
[matados2k@fortaleza curso]$ df -i
S.ficheros           Nodos-i NUsados NLibres NUso% Montado en
/dev/hda11            393600   37556  356044   10% /
/dev/hda10             26104      40   26064    1% /boot
/dev/hda6            2626560   47749 2578811    2% /home
/dev/hda8                  0       0       0    -  /home/mnt/auxi1
/dev/hda9                  0       0       0    -  /home/mnt/auxi2
/dev/hda7                  0       0       0    -  /home/mnt/juegos
none                   64333       1   64332    1% /dev/shm
/dev/hda12            904960  188632  716328   21% /usr
/dev/cdrom1                0       0       0    -  /mnt/cdrom1
[matados2k@fortaleza curso]$
```

Donde se muestra un '-' es por que ese tipo de sistema de ficheros no usan inodos (como pueden ser los FAT y los NTFS). Pero ¿por que es importante saber cuantos inodos nos quedan? Pues sencillo, si como ya comentamos que un inodo se relaciona con un fichero, si no tenemos inodos libres no pueden crearse mas ficheros por mucho espacio libre que exista.

Y por último una forma mas “humana” de usar 'df' y es usando la opción '-h', veamos el ejemplo:

```
matados2k@imperio:~$ df -h
S.ficheros           Tamaño Usado  Disp Uso% Montado en
/dev/hda6               30G   13G   16G  45% /
udev                    10M   96K   10M   1% /dev
devshm                 506M     0  506M   0% /dev/shm
/dev/hda1               16G   14G  1,1G  94% /mnt/winxp
/dev/hda7               20G   18G  2,7G  87% /mnt/juegos
/dev/hda8               29G   28G  595M  98% /mnt/auxiliar
matados2k@imperio:~$
```

Si observáis bien esta todo medido en Gigas, megas y demás que es como normalmente nosotros lo entendemos.

Cuanto tiempo llevamos 'uptime'.

uptime

Este comando simplemente informa sobre el tiempo que ha pasado desde que arrancamos nuestro Linux y la hora actual acompañada del número de usuarios y el promedio de carga que soporta el sistema. El promedio de carga significa el número medio de procesos esperando a usar el procesador. Del promedio se dan tres números, el promedio en el último minuto, en los cinco últimos minutos y en los 15 últimos minutos.

```
[matados2k@fortaleza curso]$uptime
 18:25:21  up  8:19,  4 users,  load average: 0.12, 0.18, 0.18
[matados2k@fortaleza curso]$
```

El que quiera saber mas del promedio que se dirija al siguiente enlace:
http://bulma.net/body.phtml?nIdNoticia=550

Despedida.

Hasta aquí llegamos hoy, espero que os sean útiles estos comandos y la próxima semana continuamos con mas comandos.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 11. Colección de comandos (y II).

% ar m God
ar: God does not exist

¿Quien esta aquí?, 'who'.

who [OPTION]... [FILE | ARG1 ARG2]

El comando 'who' simplemente nos muestraqué usuarios están logueados en el sistema, y desde dónde y a qué hora. Veamos su uso simple:

```
[matados2k@fortaleza matados2k]$ who
root     tty1         Jul 14 16:41
matados2 :0           Jul 14 14:46
matados2 pts/1        Jul 14 14:46
matados2 pts/2        Jul 14 16:35
matados2 pts/3        Jul 14 16:39
[matados2k@fortaleza matados2k]$
```

Sólo hay que decir que si pone 'tty**x**': '**x**' es un número que nos indica que se ha conectado desde un terminal virtual, y el número nos indica cuál de ellos (Ctrl+Alt+F1 por ejemplo). Los 'pts/**x**' son los terminales virtuales que abrimos en las X-windows (entorno gráfico). La verdad es que hay más, pero de momento no los veremos.

Como ARG1 y ARG 2 pondremos normalmente 'am I' o 'mom likes', aunque realmente puedes poner las 2 cosas estúpidas que se te ocurran, que la salida será la misma.

```
[matados2k@fortaleza matados2k]$ who am i
matados2 pts/3        Jul 14 16:39
[matados2k@fortaleza matados2k]$ who mom likes
matados2 pts/3        Jul 14 16:39
[matados2k@fortaleza matados2k]$ who yo mismo
matados2 pts/3        Jul 14 16:39
[matados2k@fortaleza matados2k]$
```

Y lo que hace es respondernos a 'who am i' o sea ¿quien soy yo? Y nos pone bajo qué usuarios estamos conectados y desde dónde hemos hecho la consulta. El parámetro FILE lo dejaremos para que uséis la ayuda y ahora veremos alguna que otra opción:

```
-a De all da información extendida o sea todo lo que 'who' nos puede dar.
-b Nos indica cuando arranco el sistema.
-d Nos da los procesos muertos o sea un historial de quien se conecto (sólo el
último por un mismo sitio).
-q Que simplemente nos da el número de usuarios.
```

Tiene bastantes más pero nos quedaremos sólo con estos. Ejemplos:

```
[matados2k@fortaleza matados2k]$ who -b
         system boot  Jul 14 14:44
[matados2k@fortaleza matados2k]$ who -d
                      Jul 14 14:44               13 id=si    term=0 salida=0
                      Jul 14 14:44             2989 id=l5    term=0 salida=0
         pts/4        Jul 14 17:02             4890 id=/4    term=0 salida=0
[matados2k@fortaleza matados2k]$ who -q
matados2k matados2k matados2k matados2k
N° de usuarios=4
[matados2k@fortaleza matados2k]$
```

Yo quiero saber más que con 'who', 'w'.

w - [-husfV] [user]

Este comando además de hacer lo mismo que 'who' nos dice qué está haciendo cada usuario, en definitiva una mezcla de 'uptime' con 'who':

```
[matados2k@fortaleza matados2k]$ w
 17:07:05  up  2:23,  4 users,  load average: 0,22, 0,12, 0,10
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
matados2 :0       -                2:46pm   ?     0.00s  0.02s  /bin/sh /usr/bi
matados2 pts/1    -                2:46pm  2:20m  0.00s  0.22s  kdeinit: kwrite
matados2 pts/2    -                4:35pm 30:59   0.07s  0.01s  man who
matados2 pts/3    -                4:39pm  0.00s  0.21s  0.01s  w
[matados2k@fortaleza matados2k]$
```

Como veis indica la hora en la que se hace la consulta, el tiempo que lleva en funcionamiento, los usuarios y la carga (acordaros de uptime) y luego nos da los usuarios con el terminal al que están conectados (TTY), desde dónde (por si es una conexión remota, FROM), cuánto hace desde que hicieron login, el tiempo ocioso (IDLE) , cantidad total de cpu usada por el usuario (JCPU), la cantidad de tiempo total usada por sus tareas (PCPU) y con que tarea está.

También podéis preguntarle por un determinado usuario pasándoselo por parámetros, y por último veremos sus opciones:

-h No imprime la cabecera (FROM,IDLE ...)

-s Da la información en el formato corto (Sin LOGIN@, JCPU y PCPU).

-f No muestra FROM.

Y como no veo necesario poner un ejemplo pasamos al siguiente.

Pues ahora quiero saber qué es ese fichero, 'file'.

file [Lista de ficheros]

Puede que tengamos archivos que son textos, otros que son binarios (ejecutables) y demás, como no es necesaria un extensión, puede que necesitemos algo para saber qué es. Esto es lo que hace 'file' concretamente, identificar el formato de un determinado fichero. Veamos unos ejemplos:

```
[matados2k@fortaleza curso]$ ls
hola_holita  lista2  nueva_lista  perro  soy_nuevo
[matados2k@fortaleza curso]$ file hola_holita
hola_holita: ASCII text
[matados2k@fortaleza curso]$ cd ..
[matados2k@fortaleza matados2k]$ file curso
curso: directory
[matados2k@fortaleza matados2k]$ cd curso/
[matados2k@fortaleza curso]$ file nueva_lista perro soy_nuevo
nueva_lista: ASCII text
perro:       data
soy_nuevo:   empty
[matados2k@fortaleza curso]$
```

Bueno, queda claro que el uso es muy sencillo y la herramienta terriblemente útil ¿o no? Este comando también tiene opciones que no comentaremos. Hay que tener bastante cuidado con este comando ya que file puede cometer errores a la hora de identificar un fichero.

Creo que voy a mutar, 'su'.

su [user]

Este comando lo que hace es convertir tu usuario actual al usuario que le indiques, por defecto con 'su' a secas lo que haremos es pedir que nos haga 'root'. Esto es de gran utilidad ya que no es necesario estar haciendo login/logout todo el rato, simplemente hay que darle la password del usuario al que queremos usar y para volver al anterior simplemente tecleamos exit o pulsamos Ctrl+D.

```
[matados2k@fortaleza curso]$ su
Password:
[root@fortaleza curso]# exit
exit
[matados2k@fortaleza curso]$ su triguelch
Password:
[triguelch@fortaleza curso]$ su
Password:
[root@fortaleza curso]# exit
exit
[triguelch@fortaleza curso]$ exit
exit
[matados2k@fortaleza curso]$
```

Evidentemente no saldrá nada en pantalla cuando tecleamos los Password, pero sí que los está recibiendo. También podéis observar cómo puedo cambiar más de una vez seguida sin hacer exit.

Despedida.

Espero que os sean útiles esta colección de comandos que os he querido dar sueltos sin meterlos dentro de una temática completa. La próxima entrega será sobre comandos de edición.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 12. Comandos de edición.

% lost
lost: not found

Más despacio por favor,' more'.

Ya hemos aprendido a visualizar el contenido de los ficheros con 'cat' , pues bien haced lo siguiente:

```
[matados2k@fortaleza curso]$ ls -l /etc > listado
[matados2k@fortaleza curso]$ cat listado
```

Y ooops! la pantalla vuela y no podemos leer lo que ponía (si estáis en entorno gráfico no vale, que tenéis barra de desplazamiento). Para hacer la visión del fichero de forma interactiva usamos lo siguiente:

more [opciones][+ número de linea][listado de ficheros]

Para empezar no veremos ninguna opción, ya que no considero ninguna muy interesante. Así que sin más pasamos a ejecutar 'more listado' (si queréis repasad el uso de pipes 'cat listado | more').

Como veis os muestra la pantalla llena de lo que contiene, y abajo lo siguiente:

```
......
drwxr-xr-x    3 root     root         4096 abr 21 03:26 CORBA
drwxr-xr-x    2 root     root         4096 jun  6  2003 cron.d
--Más-- (9%)
```

Lo que hace more es esperar a que le indiquemos una orden y nos da el porcentaje visualizado del archivo. Los comandos más comunes de espera son:

```
[barra espaciadora]   Pasa a la siguiente pantalla.
[d]   Pasa la mitad de la pantalla (mas o menos 11 lineas ya que una pantalla
normal tiene 25).
[/ expresión]   Sirve para buscar una expresión regular y dado que las
expresiones regulares pueden ser muy complejas, nosotros solamente lo usaremos
para buscar palabras (pro ejemplo '/ perro').
[Intro]  Pasa una sola línea.
[n expresión]   Como '/ expresión'  pero busca la próxima aparición de la
expresión.
[:n] Pasa al siguiente fichero de la lista en el caso que le hayamos indicado
mas de uno.
[:p] Pasa al fichero anterior.
[q] Termina la ejecución de more.
```

Bueno, no se os ocurra teclear '[' ni ']', ya que los pongo para separar lo que hay que teclear de la explicación. Otra cosa: la ejecución de more termina al llegar al 100% del último documento si no pulsamos antes 'q'.

Quiero ver el principio, 'head'.

head [opciones] [lista de ficheros]

Este comando simplemente muestra las 10 primeras líneas de los ficheros que le indiquemos:

```
[matados2k@fortaleza curso]$ head listado hola_holita
==> listado <==
total 2436
-rw-r--r--    1 root     root        15228 oct 17  2003 a2ps.cfg
...
-rw-r--r--    1 root     root          317 jul 10  2003 anacrontab

==> hola_holita <==
hola
holita
vecinitos
soy ned
[matados2k@fortaleza curso]$
```

Como opciones interesantes tiene dos, la primera es '-c' seguido del número de bytes que queremos que nos muestre (un carácter en ASCII =un byte) y la segunda el '-NUMERO' o '-n NUMERO' con lo que le indicamos el número de líneas del principio que queremos que se muestren. Ejemplos:

```
[matados2k@fortaleza curso]$ head -c 15 listado
total 2436
[matados2k@fortaleza curso]$ head -2 listado
total 2436
-rw-r--r--    1 root     root        15228 oct 17  2003 a2ps.cfg
[matados2k@fortaleza curso]$ head -n 2 listado
total 2436
-rw-r--r--    1 root     root        15228 oct 17  2003 a2ps.cfg
[matados2k@fortaleza curso]$
```

Pues ahora quiero el final, 'tail'.

tail [opciones] [lista de ficheros]

Este comando hace exactamente lo contrario que 'head', o sea mostrar las 10 últimas líneas, y lo dicho para el anterior (opciones induidas) sirve para este, así que solo veremos ejemplos:

```
[matados2k@fortaleza curso]$ tail listado hola_holita
==> listado <==
-rw-r--r--    1 root     root          289 oct 12  2003 xinetd.conf
...
-rw-r--r--    1 root     root          570 oct 29  2003 yum.conf

==> hola_holita <==
hola
holita
vecinitos
soy ned
[matados2k@fortaleza curso]$ tail -2 listado
-rw-r--r--    1 root     root          501 abr 21 03:37 yp.conf
-rw-r--r--    1 root     root          570 oct 29  2003 yum.conf
[matados2k@fortaleza curso]$ tail -c 12 listado
03 yum.conf
[matados2k@fortaleza curso]$ tail -n 2 listado
-rw-r--r--    1 root     root          501 abr 21 03:37 yp.conf
-rw-r--r--    1 root     root          570 oct 29  2003 yum.conf
[matados2k@fortaleza curso]$
```

Cuéntamelo todo, 'wc'.

wc [opciones] [listado de ficheros]

Este comando 'wc' simplemente cuenta el número de líneas, palabras y caracteres que hay en un fichero:

```
[matados2k@fortaleza curso]$ wc listado
    227    2068   15135 listado
[matados2k@fortaleza curso]$
```

Como opciones tenemos '-c' para que sólo cuente caracteres, '-l' para que sólo cuente líneas y '-w' para que sólo cuente palabras. Así que si por ejemplo usamos '-lc' nos contará las líneas y los caracteres. Si no indicamos ningún fichero nos contará lo que entremos por la entrada estándar. Ejemplos:

```
[matados2k@fortaleza curso]$ wc -c listado
  15135 listado
[matados2k@fortaleza curso]$ wc -l listado
    227 listado
[matados2k@fortaleza curso]$ wc -w listado
   2068 listado
[matados2k@fortaleza curso]$ wc -lc listado
    227   15135 listado
[matados2k@fortaleza curso]$ wc
hola
a
todos  (Pulsación de Ctrl+D)    2       3      12
[matados2k@fortaleza curso]$
```

Despedida.

Aquí llega el final de esta entrega, la próxima semana terminaremos con los comandos básicos de edición.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 13. Comandos de edición (y II).

```
% cat "food in cans"
cat: can't open food in cans
```

Busca que te busca, 'grep'.

grep [opciones] expresión [lista de ficheros]

Grep es sin duda alguna, junto con el uso de las pipes y la redirección, uno de los comandos con más utilidad de linux y por extensión unix, aunque en un principio muchos de vosotros no se la vais a encontrar. Este comando es utilizado para buscar expresiones es un fichero de texto o una lista de ellos. Veamos un ejemplo simple:

```
[matados2k@fortaleza curso]$ cat nueva_lista
aspirinas
Cerveza
Champu
Gominolas
jb
Manolito no te olvides de:
Panchitos
[matados2k@fortaleza curso]$ grep champu nueva_lista
[matados2k@fortaleza curso]$ grep Champu nueva_lista
Champu
[matados2k@fortaleza curso]$
```

Podéis observar que distingue entre mayúsculas y minúsculas, y que la salida del comando son todas aquellas líneas que contienen la expresión que queremos. Vemos con este sencillo ejemplo cómo grep es una especie de filtro que nos muestra únicamente lo que queremos, veamos otro ejemplo:

```
[matados2k@fortaleza curso]$ cat listado | grep bin
lrwxrwxrwx    1 root     root           11 abr 21 03:16 rmt -> ../sbin/rmt
drwx--x--x   93 root     bin          4096 abr 21 04:32 webmin
[matados2k@fortaleza curso]$
```

Hemos buscado dentro del listado que hicimos en entregas anteriores todos aquellos ficheros que contengan al grupo o usuario 'bin', aunque si hay una palabra que contenga 'bin' también nos la muestra.

Si necesitáramos saber en qué línea del fichero se encuentran, usaríamos la opción '-n':

```
[matados2k@fortaleza curso]$ cat listado | grep -n bin
174:lrwxrwxrwx    1 root     root           11 abr 21 03:16 rmt -> ../sbin/rmt
213:drwx--x--x   93 root     bin          4096 abr 21 04:32 webmin
[matados2k@fortaleza curso]$
```

Es evidente la gran utilidad de este comando cuando necesitemos “indagar” en un fichero de configuración ;) (cada vez queda menos para entrar en faena con cosas más interesantes) .

Para conseguir el efecto contrario de 'grep', es decir, encontrar todas las líneas donde **no** se use la expresión, usaremos la opción '-v':

```
[matados2k@fortaleza curso]$ cat nueva_lista | grep C
Cerveza
Champu
[matados2k@fortaleza curso]$ cat nueva_lista | grep -v C
aspirinas
Gominolas
jb
Manolito no te olvides de:
Panchitos
[matados2k@fortaleza curso]$
```

Para evitar que busque la expresión contenida dentro de otras palabras,véase:

```
[matados2k@fortaleza curso]$ cat listado | grep -n bin
174:lrwxrwxrwx    1 root     root           11 abr 21 03:16 rmt -> ../sbin/rmt
213:drwx--x--x   93 root     bin          4096 abr 21 04:32 webmin
[matados2k@fortaleza curso]$
```

Usaremos la opción '-w':

```
[matados2k@fortaleza curso]$ cat listado | grep -w bin
drwx--x--x   93 root     bin          4096 abr 21 04:32 webmin
[matados2k@fortaleza curso]$
```

Qué tal si corregimos las faltas ortográficas bajo consola, 'ispell'.

Este comando es un pequeño corrector ortográfico para consola, es bien sencillo de usar, así que os animo a que lo probéis sin más: “ispell nueva_lista”.

Comparemos diferencias, 'cmp'.

cmp [opciones] fichero1 [fichero2] [skip1] [skip2]

Este comando es de lo más simple de usar, simplemente compara el fichero1 con el fichero2, si éste no se indica usará 'stdin' (recordad lo del ctrl+d cuando uséis 'stdin' o entrada estándar, como prefiráis llamarlo), y muestra las diferencias. Para que lo probéis crearos una archivo con casi los mismo elementos que "nueva_lista" (a estas alturas ya deberíais ver que lo que uso una entrega lo sigo usando para las siguiente, de todas formas el contenido de ese fichero lo habéis visto en un ejemplo de esta entrega).

```
[matados2k@fortaleza curso]$ cat > nueva_lista2
aspirinas
Cocacola
Champu
Gominolas
Refresco de limon
Juanjo no te olvides de:
Panchitos
(Pulsación de Ctrl+d)
[matados2k@fortaleza curso]$ cmp nueva_lista nueva_lista2
nueva_lista nueva_lista2 son distintos: byte 12, línea 2
[matados2k@fortaleza curso]$
```

Podéis observar que nos indica dónde se produce la primera diferencia (considerar "byte" como "carácter" por si os liáis.)

No veremos opciones para este comando pero explicaré eso del 'skip1' y 'skip2', opcionalmente podemos decirle desde qué byte del documento empezar a mirar uno y otro respectivamente, y eso es lo que son estos dos parámetros opcionales. Para dónde está la siguiente diferencia haríamos lo siguiente:

```
[matados2k@fortaleza curso]$ cmp nueva_lista nueva_lista2 13 13
nueva_lista nueva_lista2 son distintos: byte 1, línea 1
[matados2k@fortaleza curso]$
```

Pero recordad que mira byte a byte (o si preferís carácter por carácter), y si hay una palabra cambiada dará una diferencia por cada carácter. Así que esto no lo hace muy útil para buscar diferencias, pero sí para saber si un fichero es idéntico o no a otro.

Para mirar bien las diferencias, 'diff'.

diff [opciones] fichero1 fichero2

Este comando es mucho más eficiente que 'cmp' a la hora de buscar diferencias, ya que lo hace por líneas y también tiene tantas opciones que lo convierte en un comando realmente complejo, por lo que lo usaremos como un 'cmp' avanzado (no veremos opciones).

```
[matados2k@fortaleza curso]$ diff nueva_lista nueva_lista2
2c2
< Cerveza
---
> Cocacola
5,6c5,6
< jb
< Manolito no te olvides de:
---
> Refresco de limon
> Juanjo no te olvides de:
[matados2k@fortaleza curso]$
```

La forma de leer la salida no es muy compleja, '2c2' significa “en la línea 2 en el segundo carácter está la diferencia”, cuando hay un '<' se refiere a lo que pone en el primer fichero indicado, y '>' al segundo. Cuando no hay diferencia no nos muestra resultado alguno.

Despedida.

Bueno ya llegamos al final, aunque quedan comandos de edición en el tintero (como 'tr', que quizás os pueda ser útil y lo explicaré si lo pedís). La próxima entrega pasaremos a ver el uso de un editor de texto bajo consola, y el porqué de saber usarlo. Un saludo.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 14. Editor de texto bajo consola.

```
% scan for <<"Arnold Schwarzenegger"^J^D
"Arnold Schwarzenegger": << terminator not found
```

¿Por qué?

Esta es la pregunta que muchos os haréis, ¿por qué aprender a usar un editor de texto bajo consola? Sobre todo cuando en entorno gráfico los hay y muy buenos (Kate, Kaedit, Kwrite, Xemacs, nedit, vim-X11, gedit .... aunque para mí ninguno como el Kwrite). Pues la cosa es bien sencilla, primero porque no todos instalaréis un entorno gráfico o no siempre tendréis un entorno gráfico (si algún día ponéis un server es un poco "tonto" gastar recursos en tener entorno gráfico), y ante cualquier problema la consola siempre estará allí (imaginaos que se os estropea las X-windows, o sea el entorno gráfico, al actualizar un driver, no queda más remedio que usar un editor para reconfigurar a mano). En definitiva, es una herramienta muy útil que nos sacará de más de un apuro, y si os convertís en usuarios habituales darle tiempo y veréis como terminái s tirando del editor bajo consola.

¿Con cuál aprenderemos y por qué?

De los muchísimos que hay y entre los más usados, vi y Emacs, nos decantaremos por este primero por una sencilla razón: no siempre puede estar instalado Emacs en un equipo (aunque raro es) y vi es casi imposible no encontrarlo, no sólo en Linux sino en cualquier tipo de Unix, sea el que sea. En definitiva, que es el único editor que está garantizado encontrar en cualquier sistema tipo Unix.

Para los que venís de la época del ms-dos veréis que vi es mucho mas 'arcaico' y más complejo de usar que el archiconocido edit, aunque con un poco de práctica lo dominaréis enseguida. Para los que queréis un editor más parecido al edit de toda la vida del ms-dos, os recomiendo el editor del Midnight Commander llamado mcedit.

Un poco de historia.

En la antigüedad los editores de texto funcionaban en modo de una línea y se usaban desde terminales tontos (o "no inteligentes", como los queráis llamar). Un editor típico que operaba en este modo es Ed. Ed era un editor potente y eficiente que consumía una cantidad ínfima de recursos (a día de hoy los recursos que consume son irrisorios) y funciona con las pantallas de la época. Vi, por decirlo así, es una alternativa visual a Ed con un grupo notablemente amplio comparado con Ed.

Vi empezó como el editor de línea de un editor llamado Ex, y éste puede verse como un modo especial de edición en vi, aunque lo contrario también es verdad (uff menudo tostón os estoy clavando hoy, si no queréis no le deis importancia a esto, simplemente considerarlo como una batallita de abuelo "cebolleta", y si no sabéis quien es éste preguntárselo a vuestros padres ;)).

Debido a lo popular de vi existen muchas variantes clónicas, y se pueden encontrar versiones para la mayoría de los sistemas operativos. Muchos de estos clones han aumentado y modificado el comportamiento original de vi y la mayoría de los clones no soportan todos los comandos originales de vi. El que vamos a usar nosotros y por ser el que viene en casi todas (por no decir todas las distribuciones de Linux) es Vim, que no es ni más ni menos que un Vi mejorado, y ni que decir tiene que lo que aprendáis para este os servirá para cualquier Vi que encontréis.

Retrocedamos en el tiempo.

Para que veáis por qué Vi es como es, veremos cómo es el Ed, aunque nunca lo vayáis a usar, yo por lo menos no lo uso y no conozco absolutamente a nadie que lo haga. Así que esta parte queda como curiosidad para que sepáis como los jurásicos en esto de la informática editaban sus ficheros.

Sin más teclead 'ed' en vuestra consola y tendréis algo como esto:

```
[matados2k@fortaleza curso]$ ed
```

Y para empezar pulsamos la 'a', que le indica que vamos a añadir texto, y para finalizar pulsaremos '.' en la primera columna de texto. Veamos el ejemplo:

```
[matados2k@fortaleza curso]$ ed
a
Esto es tan retro que voy a buscar unos pantalones de pata de elefante para mi
novia.
.
```

Para salvar el texto a un fichero se usa 'w' (una vez después del '.') seguido del nombre, y finalmente 'q' para salir del editor:

```
[matados2k@fortaleza curso]$ ed
a
Esto es tan retro que voy a buscar unos pantalones de pata de elefante
para mi novia.
.
w austin_powers.txt
79
q
[matados2k@fortaleza curso]$ cat austin_powers.txt
Esto es tan retro que voy a buscar unos pantalones de pata de elefante
para mi novia.
[matados2k@fortaleza curso]$
```

Ese 79 indica el número de caracteres guardados. Con este ejemplo tan sencillo habéis visto lo que es un modo de inserción (después de pulsar 'a') y modo de comando, concepto que nos servirá paraver a Vi.

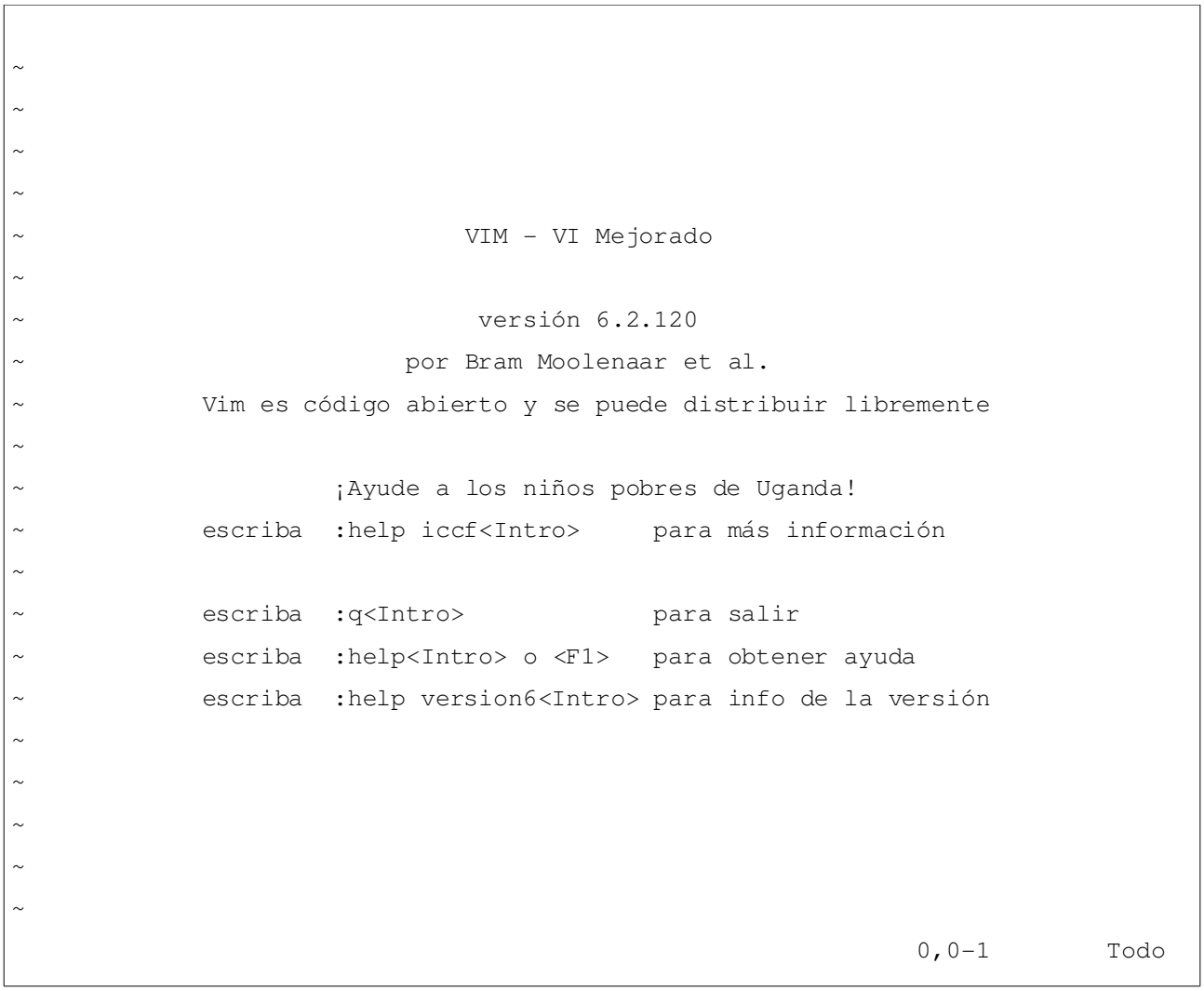
Empecemos con 'vi' a la de ya.

vi [opciones] fichero
vim [opciones] Lista_de_ficheros

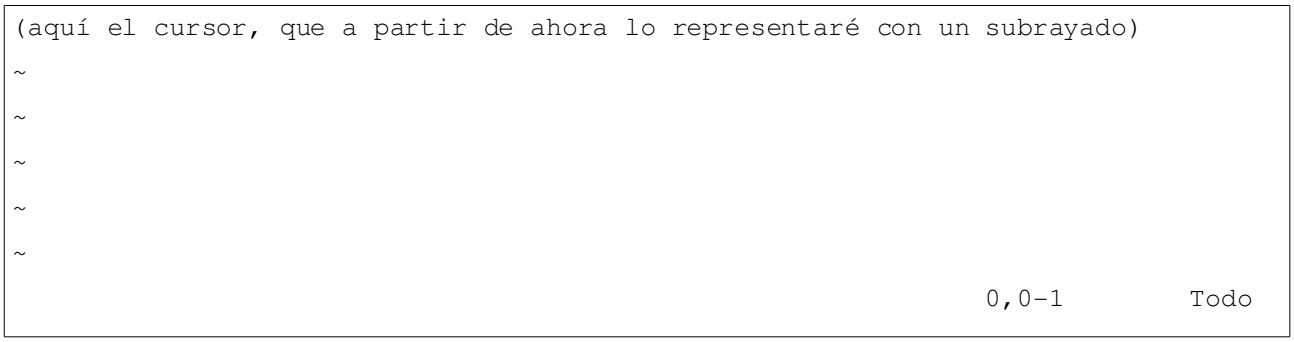
La primera es el uso del 'vi' genérico de toda la vida, y la segunda el del clon 'vim', que será el que casi todos tenemos en nuestro Linux. En todas las distribuciones (por lo menos las que yo conozco)

'vi' no es más que un enlace al ejecutable 'vim', así que nos da lo mismo teclear 'vi' que 'vim', ya que el programa que arranque será siempre el mismo, el 'vim'.

No vamos a ver opciones ni el modo de usar varios ficheros a la vez con 'vim', así que sin más tecleamos 'vim' y tendremos algo como esto:



Así que ahora pulsamos intro y empecemos a usarlo viendo algo como esto:



Donde la columna de '~' nos indica el final del texto.

Los modos de operación.

Vi tiene 3 modos de operación:

Modo comando: El modo que tenemos por defecto, que es exactamente igual al de 'ed'. Sirve para teclear nuestros comandos, que son de una sola letra.

Modo inserción: En el que estaremos la mayoría del tiempo y es el que usamos para escribir. Entramos en él pulsando 'i' y volvemos al modo comando pulsando escape.

Modo última línea: Este es un modo especial usado para dar ciertas órdenes extendidas a 'vi'. Al usar estos comandos, aparecen en la última línea de la pantalla (esto explica su nombre) Se accede a él desde el modo comando pulsando ':', y podéis usar comandos como 'q!', que sale de 'vi' sin guardar. Una vez escrito el comando pulsamos intro para ejecutarlo, y si el comando no es de salir de 'vi' volveremos al modo comando.

Todo esto parece en principio mucho lío, la verdad, y siendo sinceros un poco coñazo…pero es fácil acostumbrarse, porque al final como esta es una herramienta que usaréis la mayoría de vosotros eventualmente, usaréis 2 ó 3 comandos que recordaréis y punto.

Insertando texto.

Como ya hemos visto, para pasar del modo comando al modo inserción hay que pulsar 'i', así entraremos en el modo y podremos comenzar a escribir.

Una vez dentro de este modo, como en cualquier editor normal, podemos escribir las líneas que deseemos pulsando enter, borrar con suprimir y retroceso, y movernos por ellas con los cursores. Para salir, como ya vimos antes, pulsamos escape.

En el modo de comando también podemos movernos por el texto, y cuando pulsamos 'i' empezamos a editar justo en la posición del cursor, si pulsamos 'a' empezaremos justo en la posición anterior y si lo hacemos con 'o' lo hará una línea por debajo del cursor. ¿Es esto una chorrada? Pues sí, porque para eso tenemos los cursores y con ellos empezar donde nos plazca, así que para no complicarnos con 'i' yo creo que nos vale, personalmente yo esto lo veo solo útil en el caso de tener un teclado sin cursores o que se nos hayan estropeado, ya día de hoy creo que todos tenemos cursores, ¿no?.

```
Ya estamos por fin en faena porque matados2k eres un pesado._
~
~
~
-- INSERTAR --                                          1,61          Todo
```

Borrando texto.

Para borrar desde el modo comando usamos 'x', que elimina el carácter debajo de cursor, así que pongámonos en modo comando, vayamos a la p de pesado y pulsemos 'x' seis veces, poned lo que queráis.

```
Ya estamos por fin en faena porque matados2k eres un .
~
~
~
                                                          1,55          Todo
```

```
Ya estamos por fin en faena porque matados2k eres un cansino.
~
~
~
-- INSERTAR --                                            1,62          Todo
```

Curiosamente ;) eso es lo mismo que pulsar suprimir, así que un comando menos a recordar.

Modificar texto.

Esto es un poco pesado, ya que se hace carácter por carácter usando 'r', pulsamos 'r' junto con una letra y se cambia la letra que tengamos debajo del cursor por la nueva que pulsemos. Quizás esto no sea muy útil para cambiar una palabra entera, pero es lo que hay.

Otro comando para modificar texto más interesante puede ser '~', que cambia de mayúsculas a minúsculas y viceversa. Para eso pongámonos en la 'm' de matados2k y pulsemos 'Alt Gr + 4', que no es ni más ni menos que '~'.

```
Ya estamos por fin en faena porque Matados2k eres un cansino.
~
~
~
                                                          1,38          Todo
```

Órdenes de movimiento.

Hay múltiples órdenes para movernos por el texto, así que pondremos simplemente una tabla con ellas:

```
h   izquierda.
j   abajo.
k   arriba.
l   derecha.
w   mueve el cursor al comienzo de la siguiente palabra.
b   lo mueve a la palabra anterior.
ctrl+F   avanza una pantalla (Re Pág).
ctrl+b   retrocede una pantalla (Av Pág).
g   lleva el cursor al final del fichero.
XXg   dentro del modo última línea va a la línea XX indicada.
d$   borra todo desde la posición del cursor hasta el  final de la línea.
dG    borra todo desde la posición del cursor hasta el final del fichero.
```

Guardar el fichero y salir de vi.

Para salir de un fichero sin guardar usamos :q! .
Para guardar solo sin salir :w .
Para guardar y salir :wq .

Bueno, como veis ésta es la forma de hacerlo. Como el fichero que nosotros estamos usando no tiene nombre (ya que ejecutamos 'vi' sin mas, si ejecutas 'vi' con el nombre de un fichero, si existe lo edita y si no lo crea).

```
Ya estamos por fin en faena porque Matados2k eres un cansino.
~
~
~
:wq_
```

```
Ya estamos por fin en faena porque Matados2k eres un cansino.
~
~
~
E32: No hay un nombre de fichero                          1,17          Todo
```

```
Ya estamos por fin en faena porque Matados2k eres un cansino.
~
~
~
:wq matados.txt
```

Como veis al tenerlo sin nombre no nos deja guardarlo, así que después de ':wq' ponemos el nombre

que deseemos y ya lo tenemos.

Despedida.

En este punto vamos a dejar 'vi' con una parte pequeña de él vista, ya que nos quedan muchas cosas en el tintero como copiar y pegar, insertar otro fichero entero, cambiar a otro fichero, ejecutar una orden del intérprete de comandos desde vi, ir a la ayuda y montones de órdenes más.

Pero el objetivo de esta entrega es tener un uso básico de esta herramienta para cuando necesitemos editar un fichero de configuración y cambiar alguna opción o añadir algo, ya que sinceramente para crear ficheros normales de texto grandes puede llegar a ser muy pesado y el aprendizaje es difícil, no por ser difícil de usar en sí, sino de recordar la apabullante cantidad de comandos que necesitamos (aunque si usáis emacs sí que es para alucinar en cantidad de comandos, yo creo que con 5 dedos en cada mano no es suficiente para él ;)).

Yo por lo menos, para crear grandes ficheros, recomiendo los que son bajo entorno gráfico, tan fáciles y más potentes que el bloc de notas para un windowsero, y si aun así lo queréis en modo texto pero mas fácil os recomiendo el mcedit.

Resumiendo (¡¡¡y para eso tanto rollo!!!), para un uso eventual, cuando todo nos falla nos basta con recordar 'i' , ':q!' ,':w' y ':wq', ya que con el uso de los cursores y las teclas como borrar (retroceso), Fin, Inicio, Re Pag, Av Pág... tenemos todo lo que necesitamos, y si acaso no nos llega con esto usar ':help'.

Espero no haberos dormido con esta entrega tan larga y tostón, y sigáis con la próxima entrega donde seguramente empezaremos con la administración básica de usuarios si es que no cambio de idea ;) .

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 15. Reflexiones y mirando hacia delante.

Cuando ves la luz al final del túnel va y se te cae el techo encima.

Nota de la revisión:

Esta entrega es circunstancial y se ha mantenido como curiosidad y para mantener el número de las entregas, no aporta actualmente nada al curso, por lo que podéis pasar a la siguiente sin problemas.

Se han añadido notas para actualizar lo que se cuenta.

Reflexión.

Hasta el día de hoy, cada semana he ido poniendo una entrega por semana más o menos puntual, pero siempre una por semana, menos cuando empezó que puse tres de golpe y ahora que pongo 2 en una misma semana (En la revisión se están poniendo entre 6 y 7 por semana, hasta que la revisión termine y se publique 1 por semana, cosa que pasará después de la entrega número 30). Y por cierto, tengo que decir que estoy muy contento con la audiencia que está teniendo, que yo la estimo en más de 100 personas que lo siguen fielmente (Como se paró el curso durante mucho tiempo, esto ya puede no ser verdad, pero seguro que lo recuperaremos en cuando Sinuh y Noticias3d se actualicen, ya que hay problemas actualmente y tardarán algo más). La verdad, nunca me esperé tanto, tanto es así que me siento animado a continuar durante mucho más tiempo.

Incluso se me ha pasado por la cabeza hacer otro curso sobre programación en entornos Unix/Linux también desde 0 (Esto es ya una realidad), pero mi tiempo, sobre todo porque quiero terminar ya mi proyecto para acabar la carrera (Actualmente soy programador de sistemas de información geográfica entre otras cosas, vamos, chico para todo), no es excesivamente abundante y no voy a dejar de hacer vida social ;) (curiosamente hay vida después de todo esto XD). ¿Quizás cuando lo termine? Pues, puede.

También da gusto ver cómo si pones en google "curso GNU/Linux" sale el primero de 70 páginas de resultados (unos 682) con tan poco tiempo de vida, y juro que no he hecho ningún conocido truco (Esto actualmente ya no es cierto).

La participación el foro para el curso se puede decir que es prácticamente nula y curiosamente preguntáis donde está habilitado para hacer comentarios, y claro, esperar respuestas ahí pues como que es un poco difícil. Por cierto, gracias por los ánimos en los comentarios. Espero que la causa por la que no usáis los foros al menos sea porque me explico muy bien o porque aún estamos en lo más básico.

Tampoco he recibido ni un solo e-mail con ninguna petición sobre algún tema, ni ofreciéndose para hacer complementos a las entregas para el que quiera saber más, ni sugerencias, ni comentarios... supongo que lo leído en la entrega 0 os echa para atrás de hacerlo, aprovecho para animaros a que lo hagáis, sobre todo para las sugerencias y peticiones (Actualmente los mails recibidos se pueden contar con los dedos de una mano).

En definitiva, estoy muy contento porque "creo" que os gusta, y esto lo deduzco por las visitas que tiene, espero que siga durante mucho tiempo más (Actualmente se superan las 125000 visitas en suma total de entregas en sólo en Sinuh).

Mirando hacia delante.

Ahora que ya está visto el uso mas básico de la consola empezaremos con cosas un poco más complejas como pueden ser la administración de usuarios, ficheros de configuración, redes ... así que, dependiendo de lo complejo que sea el tema que toque en cuestión, puede que aumente a quincenal la aparición de la entrega, cosa que intentaré ir avisando al igual que avisaré si una semana o una quincena no vaya a salir una entrega, por ejemplo porque me vaya de viaje, cosa que lamentablemente (para mí) no hago mucho, sobre todo por aquellos que siguen puntualmente las entregas.

En lo referente a si seguiremos machacando la línea de comandos, la respuesta es que sí y por una razón muy sencilla que cae por su propio peso: dada la gran cantidad de distribuciones que hay y que no hay una clara "gran mayoritaria", y que cada una implementa sus herramientas, como por ejemplo la de la próxima entrega "gestión de usuarios", donde Red-hat tiene una herramienta, Suse otra, Mandrake la suya... , es imposible ver cada una de ellas y lo que siempre vas a tener garantías en toda distribución es que bajo consola en todas se hace igual (excepciones las hay, como en todo), y lo que aprendas en una te servirá para cualquiera, y es más, los conceptos que aprendas te servirán para cualquier herramienta gráfica que encuentres para hacer lo mismo (Esto sigue siendo cierto).

¿Quiere decir eso que no veremos entornos gráficos? En absoluto, a partir de ahora es cuando empezarán las entregas en las que veremos herramientas para ambos entornos, tanto en línea de comandos como bajo entornos gráficos, aunque tenedlo claro: primero será bajo la consola y luego en entorno gráfico en la mayoría de los casos.

Más o menos tengo una idea de por dónde voy a seguir las entregas, pero ahora es cuando más agradecería que hicierais peticiones para saber qué os interesa más. Pero comprended que no por hacer una petición vaya a salir una entrega inmediata o próximamente sobre ese tema por dos motivos: hay que intentar ir aumentando la dificultad y los conocimientos gradualmente, con esto quiero decir que si por ejemplo a día de hoy me pedís una entrega sobre la compilación del kernel pues vais a tener que esperar, porque hay que ver antes cosas mucho más básicas. Y la segunda razón es que yo no lo sé todo, evidentemente, y me considero muy lejos de ser un gurú de GNU/Linux y por extensión de Unix, y hay temas que puede que no domine o que tenga que consultar antes.

También intentaré ya por fin hacer las revisiones de las primeras entregas y publicarlas en breve (Esto es lo que se esta haciendo a día de hoy).

Despedida.

Creo que ya no se me olvida nada más del tostón que quería soltaros, así que os dejo hasta la próxima entrega. Un saludo a todos.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 16. Gestión de usuarios.

$ nice man woman
No manual entry for woman

Introducción.

Aunque tú seas el único usuario dentro de tu sistema, se debe tener una cuenta de root. Sin embargo, no es nada aconsejable usar siempre root como si fuéramos un usuario de Windows, es aconsejable tener también una cuenta de usuario normal. Si optas por usar siempre root por comodidad o porque no quieres adaptarte a un uso más seguro del sistema… tú mismo, pero te arriesgas a cargarte el sistema fácilmente, ya que puedes hacer "todo" y muchas veces el sistema ni te pedirá una confirmación, y ante un ataque darás al atacante más facilidad ya que, si consigue que se ejecute código en tu máquina, ese código tiene vía libre a lo que quiera hacer.

Lo mejor es usar root simplemente cuando te es imprescindible (como cambiar una configuración) y luego vuelve a tu usuario normal, esto ya lo vimos con el comando 'su'. Incluso si te logueas como root en entorno gráfico, cuando termines sal y loguéate como uno normal.

Conceptos de gestión de usuarios.

El sistema guarda unos datos relativos acerca de cada usuario, y dicha información se encuentra en el fichero /etc/passwd. Cada línea contiene información acerca de un usuario en el siguiente formato:

nombre:clave_encriptada:UID:GID:GECOS:directorio_inicial:intérprete

El primer campo "nombre" es el nombre del usuario, seguidamente está la clave encriptada aunque lo que todos encontraréis seguramente es una 'x', y dentro de poco veremos por qué. Los ':' son simplemente un separador entre los campos.

Después nos encontramos con el UID y GID, el primero es el identificador de usuario para el sistema y el segundo el identificador de grupo primario al que el usuario pertenece, puesto que un usuario puede pertenecer a muchos grupos.

GECOS es un campo opcional para guardar información. Normalmente contiene el nombre completo del usuario. GECOS significa General Electric Comprehensive Operating System. En este campo podemos poner lo que queramos ya que el sistema no lo usa para ninguna gestión. Se puede omitir sin poner entre los ':' nada, quedando '::'.

El directorio inicial es donde empezará el usuario una vez logueado, y el intérprete indica cuál interprete de comandos usará (bash, sh, tcsh...). Si en el campo intérprete encontramos '/sbin/nologin' es un usuario que no puede loguearse, esto tiene sentido ya que el propio sistema tiene usuarios con privilegios para dar servicios al sistema como pueden ser 'bin', 'daemon', 'adm'...

El usuario root siempre tiene como UID y como GID un '0', y cualquier usuario que lo tenga tendrá los mismos privilegios que él. Hay que recordar que un usuario sólo puede tener un UID, pero varios usuarios pueden tener el mismo UID, cosa totalmente desaconsejable porque puede provocar agujeros de seguridad importantes. Si ves un usuario con UID=0 que no sea root, lo más seguro es que hayas sido atacado por algún, ¿hacker?, bastante cutre.

Normalmente el sistema guarda los UID bajos (por ejemplo por debajo de 500 o por debajo de 1000) para los usuarios del propio sistema, los que necesita para los servicios que ofrece (por ejemplo, si tienes un servidor de ftp lo normal es tener un usuario en el sistema que se llame 'ftp'), y los UID altos se utilizan para los usuarios normales.

Si donde debiera aparecer la clave aparece '::', esto indica que no hay clave y por supuesto no es nada bueno. Si aparece una 'x' es que las claves o la clave de ese usuario están gestionadas en el archivo /etc/shadow ya que la mayoría de los sistemas usan "claves en sombra". Esto es debido a que /etc/passwd debe ser visible a todos, porque si no ciertos comandos como puede ser 'ls' dejarían de funcionar, y aunque la clave está encriptada no es bueno que se encuentre a la vista por la existencia de programas de fuerza bruta que se dedican a desencriptarlas (el famoso jhon deriper por ejemplo), y cualquier usuario con acceso a nuestro sistema tendría las claves usando dichos programas. En vez de esto se usa el fichero /etc/shadow, que sólo es visible por root.

Como no es aconsejable para nada tocar estos ficheros manualmente, no continuaremos con la descripción del fichero shadow (quien quiera verlo que use 'man 5 shadow'), en lugar de tocar los ficheros usaremos los comandos que nos permiten manejar todo esto.

Añadir usuarios.

useradd [-c comentario] [-d home] [-e fecha] [-f dias] [-g grupo] [-G lista de grupos] [-m [-k template] | -M] [-n] [-o] [-p passwd] [-r][-s shell] [-u uid] usuario

No os dejéis asustar por el tamaño de este comando, podéis usar 'useradd' o 'adduser' para añadir usuarios al sistema, puesto que es el mismo comando. Primero pasaremos a comentar cada una de las opciones.

```
[-c comentario]   Pondremos el comentario que queremos en el campo GECOS.
[-d home]         Directorio home (el de inicio) de la cuenta.
[-e fecha]        fecha en formato año-mes-día en que la cuenta caduca y se
                  bloquea.
[-f dias]         Número de días en los que la cuenta se bloqueará si no se
                  usa.
[-g grupo]        Nombre del grupo primario. Ojo el grupo debe ya existir.
[-G lista de grupos]     Listado de grupos al que el usuario pertenecerá.
[-m [-k template] | -M]  -m Crea el directorio home de la cuenta si es que no
                         existe ya. Con -k usa el directorio template para
                         crear el directorio home o sea que copia lo de
                         template al home, en caso contrario se copia /
                         etc/skell. Si en vez de -m ponemos -M el directorio no
                         es creado.
[-n]              Añade un grupo con el mismo nombre que el usuario
[-o]              Permite crear usuarios con uid duplicada.
[-p passwd]       Añade el password al usuario.
[-r]              Se usa para crear un usuario del sistema.
[-s shell]        Indica que shell
[-u uid]          Indicamos que uid queremos.
```

¿Por qué no asustarse con tanto comando? Pues bien, si no indicamos nada el 'home' del usuario se crea por defecto con el '/etc/skell', será un usuario indefinido y no se bloqueará por no usarse, tendrá la shell por defecto (normalmente bash), se le asigna un uid automaticamente y crea un grupo con el mismo nombre y se lo pone como principal. Por lo que ejecutar 'useradd -m -n -u el_siguiente_uid_no_ocupado -s /bin/bash nuevo_usuario' es lo mismo que 'user_add nuevo_usuario'

así que a menos que queramos cambiar el funcionamiento por defecto, no tenemos porqué complicarnos.

```
[root@fortaleza root]# useradd petardo
[root@fortaleza root]# useradd petardo -p petardeitor
useradd: user petardo exists
[root@fortaleza root]#
```

Como podemos ver en el ejemplo, no se puede usar useradd para luego añadirle algo que nos falte al usuario que creamos. Nota: Este comando, al igual que el resto, sólo los puede ejecutar 'root' ,y si usáis el entorno gráfico al utilizar 'su' hacerlo con la opción '-l', porque si no entonces no encontrará el comando, esto es así por que 'su' no cambia el path del usuario por el del root, y hay que indicar la opción '-l' para que así sea (realmente lo que hace es que carga todo el entorno del root, pero de momento no os preocupéis si no lo entendéis).

Como veis es mucho más cómodo usar el comando que añadirlo todo a mano usando los ficheros de configuración. Os preguntaréis qué es '/etc/skell', este directorio es como un directorio 'modelo', el cual se copiará al home de los nuevos usuarios creados, y es aquí donde podéis poner todo lo que queráis que tengan todos los usuarios nuevos de vuestro sistema.

Modificando al usuario.

Como visteis antes no se pueden cambiar las características de un usuario con useradd (oadduser), hay que hacerlo con el siguiente comando:

usermod [-c comentario] [-d home] [-e fecha] [-f dias] [-g grupo] [-G lista de grupos] [-m] [-n] [-p passwd] [-s shell] [-u uid [-o]] usuario [-L | -U] usuario

El significado de los parámetros es el mismo que en el anterior comando, así que no necesitan explicación, salvo dos nuevos:

```
-L Bloquea la cuenta de usuario.
-U La desbloquea.
```

Así que vamos a añadirle un password a nuestro usuario petardo y bloquearlo y desbloquearlo:

```
[root@fortaleza root]# usermod -p petardeitor petardo
[root@fortaleza root]# usermod -L petardo
[root@fortaleza root]# exit
[matados2k@fortaleza curso]$ su petardo
Password:
su: contraseña incorrecta
[matados2k@fortaleza curso]$
```

Como veis no nos deja loguearnos como petardo, puesto que lo tenemos bloqueado.

¿Yo puedo cambiarme mi contraseña?

Imagínate que no estás en un ordenador tuyo, sino que estás en un sistema en el que tienes una cuenta de usuario, sería una molestia no poder cambiar tu propia contraseña y tener que pedírselo al administrador (aparte de no gustarte que el administrador sepa tu clave). Pues para esto el siguiente comando:

passwd [opciones] [usuario]

No veremos ninguna opción y como podréis probar el único que puede cambiar una contraseña que no sea suya es 'root'.

```
[matados2k@fortaleza curso]$ su -l
Password:
[root@fortaleza root]# usermod -U petardo
[root@fortaleza root]# exit
[matados2k@fortaleza curso]$ su petardo
Password:
[petardo@fortaleza curso]$ passwd
Changing password for user petardo.
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[petardo@fortaleza curso]$
```

Como veis nos pide el viejo password y 2 veces el nuevo (para no equivocarnos). Puede que no nos deje poner una clave algo sencilla, si quieres hacerlo usa 'passwd' con la opción '-o'.

Despedida.

En la próxima entrega veremos cómo eliminar a los petardos de nuestro sistema ;) y cómo administrar los grupos.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 17. Gestión de usuarios (y II).

$> cd /pub
$> more beer

Eliminando al petardo, digo… eliminando usuarios.

Como creamos en la anterior entrega un usuario que era petardo, pues ya es hora que lo eliminemos, y para eso existe el siguiente comando:

userdel [-r] usuario

Simplemente elimina al usuario que le indiquemos (mirad qué sencillo esta vez, ya se sabe que destruir es más fácil que crear), y como única opción tenemos '-r', que elimina su directorio home con todos sus ficheros.

Así que ya podemos eliminar todos los petardos de nuestro sistema:

```
[matados2k@fortaleza curso]$ su -l
Password:
[root@fortaleza root]# userdel -r petardo
[root@fortaleza root]# ls /home/petardo
ls: /home/petardo: No existe el fichero o el directorio
[root@fortaleza root]# exit
[matados2k@fortaleza curso]$
```

Y con los petardos eliminados sin rastro alguno vamos a pasar a los grupos.

Ahora toca a los grupos.

Como ya vimos en la entrega pasada y en otra anterior (de cabeza no puedo acordarme de todas :P), cada usuario pertenece a uno o más grupos (aunque sólo uno es el principal), y por eso cada fichero tiene un usuario y un "grupo propietario" (el principal del usuario), por decirlo así.

Al igual que hay usuarios del sistema, también existen grupos del sistema tales como grupo root, grupo bin… y por lo general, salvo raras excepciones, los usuarios **nunca** deben pertenecer a ninguno de esos grupos.

La información de los grupos se guarda en '/etc/group', y al igual que los usuarios también pueden tener y es normal tener las claves bajo shadow, concretamente el fichero de shadow para los grupos es '/etc/gshadow', e igual que en los usuarios solo es accesible por el root. Del mismo modo que con los usuarios solo explicaré el contenido del primero.

El formato de '/etc/group' es el siguiente:

nombre_grupo:clave:GID:lista de usuarios miembros

El primer y el segundo campo no necesitan explicación ya que es igual que en lo de usuarios (acordaos que si hay una x es que la clave está bajo shadow). GID es el identificador de grupo y suele coincidir que el UID del usuario con el mismo nombre, pero no tiene porqué serasí. Y por último tenemos la lista de todos los usuarios que pertenecena ese grupo.

No es normal que los grupos de usuarios tengan clave, si conocéis cómo es '/etc/gshadow' veréis que el campo de la clave esta vacío (de hecho, aun sin saberlo como root, miradlo y veréis que casi todos los campos están en blanco menos el primero y el último), no se suelen usar nunca las claves en los grupos (la verdad es que sería muy agobiante que te pidiera la clave de grupo también).

Añadiendo grupos.

groupadd [-g gid [-o]] [-r] [-f] group

Como veis esto ya se simplifica con respecto a lo usuarios. Así que pasemos a ver qué es cada opción:

```
-g indica explícitamente el GID del grupo.
-o no obliga a que el identificador de grupo sea único, cosa totalmente desaconsejable.
-r para crear un grupo del sistema.
-f hace que groupadd no de error si el grupo ya existe.
```

Así que vamos a crear un grupo llamado, por ejemplo, botijo:

```
[matados2k@fortaleza curso]$ su -l
Password:
[root@fortaleza root]# groupadd botijo
[root@fortaleza root]# groupadd -f botijo
[root@fortaleza root]# groupadd botijo
groupadd: group botijo exists
[root@fortaleza root]# exit
[matados2k@fortaleza curso]$
```

Cambiando las características de los grupos.

Hoy estamos caprichosos y queremos modernizarnos, queremos cambiar el grupo botijo por cantimplora que es más portátil XD (No señores, no he fumado cositas de esas raras que se le echan al tabaco). Para eso necesitamos el siguiente comando:

groupmod [-g gid [-o]] [-n group_name] group

Otra vez vemos que es sencillo y que el significado de las opciones es el mismo, menos '-n' que es para cambiar el nombre, opción que demostraré con el ejemplo:

```
[matados2k@fortaleza curso]$ su -l
Password:
[root@fortaleza root]# groupmod -n cantimplora botijo
[root@fortaleza root]# groupadd botijo
[root@fortaleza root]# exit
[matados2k@fortaleza curso]$
```

Como podéis observar podemos volver a crear el grupo botijo, ya que al renombrarlo deja de existir para llamarse cantimplora.

Eliminado los grupos.

Para eliminar los nuevos grupos que hemos creado usamos el sencillo comando:

groupdel group

Con este simple comando que no tiene ni opciones eliminaremos los grupos:

```
[matados2k@fortaleza curso]$ su -l
Password:
[root@fortaleza root]# groupdel botijo
[root@fortaleza root]# groupdel cantimplora
[root@fortaleza root]# exit
[matados2k@fortaleza curso]$
```

La guinda.

Para terminar la gestión de usuarios veremos dos comandos que nos pueden ser útiles:

users
groups usuario

El primero indica los usuarios que están accediendo al sistema (acordaos del comando 'w'), y el segundo indica todos los grupos a los que pertenece un usuario. Así que nada más que un ejemplo:

```
[matados2k@fortaleza curso]$ users
matados2k matados2k matados2k
[matados2k@fortaleza curso]$ groups matados2k
matados2k : matados2k
[matados2k@fortaleza curso]$
```

Despedida.

Con esto ya terminamos esta parte y espero que no os haya aburrido mucho. En la próxima entrega empezaré con cómo se instalan los programas en GNU/Linux. Intentaré empezar con los programas que deben ser compilados, después con los rpm,luego con los deb y finalmente con el apt-get. Hasta la próxima.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 18. Instalando programas (I).

```
% alias alias alias
alias: Too dangerous to alias that.
```

Cómo va eso de instalar.

En Linux, como en otros Unix, los programas no se suelen instalar con siguiente, siguiente, siguiente, OK... como pasa en Windows, aunque sí existen programas que se instalan de esta forma, como nuestro querido amsn (programa para conectarte a la red Microsoft Messenguer, con una interfaz muy parecida al msn de siempre), con su instalador y muchos de los programas comerciales (incluido juegos). Pero por desgracia son minoría.

Básicamente, descartando lo primero, hay dos formas de instalar los programas bajo Linux, con los fuentes y los binarios. Una de las grandezas del software libre es la posibilidad de disponer del código fuente, el cual siempre podrás modificar según tus necesidades o hacer que terceros lo hagan por ti. Aun siendo la forma más difícil de instalar un programa, y seguramente la que más veces huiréis de ella, es sin duda la que puede traernos mayores beneficios.

¿Pero qué beneficios tiene instalar a partir del código fuente? Uno de los beneficios más inmediatos es la posibilidad de compilarlo en otras arquitecturas, sí señores, hay vida más allá de la arquitectura del pc, y lo que más te puede interesar es que puedes optimizar el programa para tu máquina (hay distribuciones que se basan en esto como Gentoo, por si eres un fan de la optimización). Como en Windows, los programas vienen generalmente compilados para funcionar en la mayoría de ordenadores posibles (generalmente, compiladas desde 386 a Pentium II), lo que hace que no aprovechemos muchas de las capacidades y/o ventajas de nuestros procesadores modernos, aunque también es cierto que muchas veces ni se nota mejoría cuando está un programa compilado específicamente para nuestro procesador. Es como todo, depende, ¿de que depende?, pues de muchas cosas ...

Y la otra forma básica son los binarios, que son el resultado de la compilación anterior y su ventaja es que nos facilita la vida ahorrándonos el paso anterior y hacérnoslo todo un poco más cómodo. Los binarios generalmente vienen empaquetados (es difícil no encontrarlos así), y estos paquetes tienen a día de hoy dos formatos, los paquetes rpm y los paquetes deb. Los primeros son originales de Red Hat y han sido adoptados por multitud de distribuciones, como pueden ser Suse o Mandrake/Mandriva, y el LSB (Linux Standard Base) los define como el estándar de paquetes. Y luego están los paquetes deb, que son igual de buenos o mejores que los rpm y son originales de la distribución de Debian, y la suelen utilizar aparte de ésta todas las distribuciones basadas o descendientes de Debian que, todo hay que decirlo, no son pocas.

Pero no todo queda aquí, dadas las diferencias entre distribuciones muchas veces los paquetes compilados para una distribución no sirven para otra, o hay que hacer chapucillas de por medio y esto es lo que suele pasar con los rpm, por lo que es recomendable instalar solamente paquetes compilados específicamente para tu distribución, si bien es verdad que se pueden instalar de unas a otras sin problemas muchísimas veces. Otros paquetes simplemente no van marcados para ninguna distribución en concreto y funcionan en todas. Al final todo se reduce a prueba y error, si no encuentras un paquete compilado para tu distribución, prueba con la del vecino, que alguna funcionará. Este problema no los suelen tener las distribuciones basadas en Debian, que son más parecidas internamente entre ellas.

Pero ahora vamos a rizar el rizo, porque también hay distribuciones que funcionan con ambos sistemas de paquetes como Yoper, y si tu distribución no es de éstas (que es lo más normal) siempre puedes alienar los paquetes del otro formato, esto es, convertir un rpm en un deb y viceversa, esto se hace con un programa que veremos en otra entrega, llamado alien.

¡Oh, no! Dependencias.

Pero por desgracia aquí no queda todo, y esto es lo que menos os va a gustar. En linux se lleva al máximo la reutilización del código (Si algo funciona y funciona bien ¿para qué volver a reinventar la rueda?), lo que hace que unos programas se apoyen en otros o en librerías, con lo cual se nos devuelve un error de dependencia si intentamos instalar/compilar un programa que dependa de otro u otra librería y no tenemos el susodicho en nuestro sistema.

La solución es bien sencilla, ¿no? ¡Instalemos lo que nos falta! Pero si lo que nos falta depende a la vez de otros, y esos otros de otros, pues acabarán con nuestra paciencia y, por desgracia, esto es más común de lo que parece, así que cuanto más grande es un programa su tendencia a tener dependencias aumenta.

Y lo peor aún y que puede ocurrir es, por ejemplo, que actualizamos a una versión más moderna de un programa que utilicemos y resulta que las dependencias que cumplían antes bien ya no valen, porque como todo, todo evoluciona y van saliendo nuevas versiones, y claro... toca volver a instalar dependencias, con lo cual nuestra tendencia a buscar un martillo aún más gordo para pagar nuestra desesperación con el ordenador aumenta progresivamente.

Al fin, un poco de luz al final del túnel.

Pero tranquilo, vuelve a por los CD's de las distribuciones que acabas de tirar por el balcón y ¡tú! Manolito (los Manolitos que se sientan ofendidos por usar su nombre, pueden cambiarlo por otro XD), deja ese tenedor donde estaba, no hace falta que crees un nuevo CD de música alternativa.

Todo puede hacerse mucho más fácil con los gestores de paquetes (y no me refiero a rpm para rpm y dkpg para deb), que resuelven las dependencias por nosotros y se descargan (o buscan en los CD's, según el caso) lo necesario para que todo funcione (y de esto saben mucho los de Debian), así sin más, sin trucos, aunque aun así, como bien Murphy sabe, todo puede fallar, porque ya lo puse en una entrega: "cuando ves la luz al final del túnel, va y se te cae el techo encima", pero esto sólo suele pasar cuando instalamos de donde no debemos, como por ejemplo de repositorios de ramas inestables.

¿Y qué son los repositorios? Pues son "sitios" en Internet donde nuestro maravilloso gestor buscará y descargará los paquetes necesarios o simplemente los CD's de nuestra distribución favorita, ya sea bien para instalar software como para actualizarlo, ¡incluso la distribución entera! Y es que esto de que sea software libre aunque cueste adaptarse tiene sus ventajas, vamos a hacer lo mismo con software propietario a ver qué nos dicen }:D.

Y en este arte de los repositorios los que se llevan la palma son los de Debian, que prácticamente todo lo pueden encontrar en sus repositorios, porque está claro que no siempre vamos a encontrar lo que queríamos de esta forma tan cómoda, y tendremos que buscar e instalar a mano, e incluso a veces sólo tendremos los fuentes y tocará compilar.

¿Por dónde empezamos?

Y con esto anterior ya leído, y como deduciréis, lo veremos todo en el orden que lo he ido comentando, así que empezaremos por tener que compilar, después por la gestión de paquetes rpm,

luego la de paquetes deb y terminaremos explicando el uso de un gestor de paquetes y ese es apt-get, sí, el original de Debian, por la sencilla razón de que puede usarse independientemente del tipo de paquetes y porque existen ports (vamos, que se han "portado") para multitud de distribuciones. Hay más dependiendo de la distribución que uséis, como yum para Red hat , yast (de hecho yast es mucho más que eso) para Suse o urpmi para Mandriva, pero no los veremos porque como siempre intentaremos que esto sirva para el mayor número posible de gente.

Y ya puestos... ¿qué necesito para compilar?

Pues fácil, el compilador. ¿Qué compilador? Pues depende de los fuentes, pero podéis estar seguros de que el 90% de los casos será C o C++, y para eso y mucho más tenemos a los maravillosos compiladores de la GNU que cumplen las 3 b's: bueno, bonito y barato. Para C y C++ será el gcc, y viene en absolutamente todas las distribuciones, así que si no lo tenéis ya podéis meter el CD y me juego la cabeza que está en el CD nº1, e instaladlo con la ayuda del instalador de paquetes propio de vuestra distribución (o tendréis que esperaros a que llegue la entrega del apt-get), y aquí no puedo ayudaros porque cada una tiene el suyo (en una Fedora estará en Menú inicio -> sistema -> Añadir/quitar aplicaciones, en una Mandriva dentro de la herramienta de configuración de Mandriva, y así con cada una).

Si no sabéis si lo tenéis o no, teclead en consola lo siguiente:

```
[matados2k@fortaleza matados2k]$ cc
cc: no hay ficheros de entrada
[matados2k@fortaleza matados2k]$ gcc
gcc: no hay ficheros de entrada
[matados2k@fortaleza matados2k]$
```

Si sale algo parecido a esto lo tenéis, con que ejecutéis uno de los dos vale. Si en cambio pasa esto:

```
[matados2k@fortaleza matados2k]$ cc
bash: cc: command not found
[matados2k@fortaleza matados2k]$ gcc
bash: gcc: command not found
[matados2k@fortaleza matados2k]$
```

Ya sabéis lo que toca :(.

Aunque no programéis o no tengáis ni idea de C/C++ conviene que lo instaléis, porque C está tan ligado a Linux como una madre a su hijo, de hecho C se hizo para reescribir Unix y Linux no deja de ser una implementación libre de Unix, y ya veréis como tarde o temprano tendréis que usarlo, aunque sea indirectamente.

Otros compiladores que recomendaría que tuvierais son los de perl, python y java que son también muy usados. Y no os olvidéis del intérprete de tcl, que seguro que también tarde o temprano lo necesitaréis (aunque seguro que muchos por defecto lo tenéis). Y tampoco tenéis que preocuparos por no saber lo que son, simplemente quedaos con sus nombres. Solo un apunte más, en el caso de Java usad mejor el de Sun que el de GNU, ya que evitaréis muchos problemas.

Manos a la obra.

Generalmente, los fuentes vienen empaquetados conextensiones .tar, .tar.gz, gz, bz2, tar.bz2 así que pondré una tabla de cómo descomprimir cada uno de ellos, porque aúnno toca entrega de compresores/descompresores(aunque ya creo que va a ser lo próximo después de lo de instalar) .

```
tar (.tar)          tar -xvf archivo.tar
gzip (.gz)          gzip -d archivo.gz
bzip2 (.bz2)        bzip2 -d archivo.bz2
(tar.gz)            tar -xvzf archivo.tar.gz
(tar.bz2)           bzip2 -dc archivo.tar.bz2 | tar -xv
```

Y para practicar vamos a compilar un juego muy entretenido llamado supertux, lo podéis encontrar en http://super-tux.sourceforge.net/ , en el momento de escribir esta entrega nos bajamos laversión 0.1.2 en su formato de fuentes supertux-0.1.2.tar.bz2 y descomprimimos:

```
[matados2k@fortaleza curso]$ bzip2 -dc supertux-0.1.2.tar.bz2 | tar -xv
supertux-0.1.2/
supertux-0.1.2/contrib/
.....
supertux-0.1.2/data/Makefile.in
supertux-0.1.2/data/Makefile.am
[matados2k@fortaleza curso]$ supertux-0.1.2
[matados2k@fortaleza supertux-0.1.2]$
```

No hay un método fijo por el cual hacerlo, ya que eso depende del programador de la aplicación en cuestión, pero sí que hay una forma muy implantada de hacerlo y suele ser la siguiente:

- 1º Si no existe un archivo llamado 'configure' lo creamos con el script 'autogen.sh', si es que éste existe. Y aunque exista 'configure'conviene generarlo de nuevo (a mi se me ha dado el caso de que hasta que no he usado el 'autogen.sh' no era capaz de compilarlo).
- 2ª Ejecutar 'configure', y es en este punto donde nos dirá si nos falta algo parándose conun error, hasta que no se satisfaga 'configure' no se tendrá una compilación fructífera.
- 3ª Ejecutamos 'make' (comando que es necesario instalar y que suele venir ya instalado, usa el fichero 'makefile' que se genera despues del 'configure').
- 4º Ya como root instalamos la aplicación con 'make install'.
- 5ª La desinstalación generalmente se suele hacer, dentro del mismo directorio donde se compiló y siendo root con 'make uninstall' o 'make clean' (siendo este último casi siempre para limpiar los rastros de una antigua compilación).

Pero como no hay regla fija para esto, siempre hay que leerse los ficheros tipo LEEME o INSTALL, o la documentación de la página web donde se encontraba el programa, donde nos indicarán todos los pasos exactos y que necesitamos.

En nuestro ejemplo necesitaremostener instaladas las librerías SDL, SDL_mixer y SDL_image (Opcionalmente OpenGL) queencontraréis seguramente en los CD's de vuestra distribución. Yo como las tengo voy manos a la obra:

```
[matados2k@fortaleza curso]$ cd supertux-0.1.2
[matados2k@fortaleza supertux-0.1.2]$ ./autogen.sh
[matados2k@fortaleza supertux-0.1.2]$ ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
......
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating data/Makefile
config.status: executing depfiles commands

Features:
=========
 Profile Mode:   no
 Debug Mode:     yes
 OpenGL Support: yes

[matados2k@fortaleza supertux-0.1.2]$
```

Como veis ya tengo los pasos 1 y 2 hechos, y como resultado me crea los 'makefile' necesarios y me indica que se compilará con OpenGL ya que tengo las librerías necesarias.

```
[matados2k@fortaleza supertux-0.1.2]$ make
Making all in src
make[1]: Cambiando a directorio `/home/matados2k/curso/supertux-0.1.2/src'
....
make[1]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2/src'
Making all in data
make[1]: Cambiando a directorio `/home/matados2k/curso/supertux-0.1.2/data'
make[1]: No se hace nada para `all'.
make[1]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2/data'
make[1]: Cambiando a directorio `/home/matados2k/curso/supertux-0.1.2'
make[1]: No se hace nada para `all-am'.
make[1]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2'
[matados2k@fortaleza supertux-0.1.2]$
```

Como veis no me da ningún error, y después de ver volar texto sin parar ya tengo mis binarios, que incluso puedo usar sin instalarlo (entrando en '/src' tenemos el ejecutable 'supertux'). Ahora pasamos a instalarlo.

```
[matados2k@fortaleza supertux-0.1.2]$ su
Password:
[root@fortaleza supertux-0.1.2]#make install
.....
make[2]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2/data'
make[1]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2/data'
make[1]: Cambiando a directorio `/home/matados2k/curso/supertux-0.1.2'
make[2]: Cambiando a directorio `/home/matados2k/curso/supertux-0.1.2'
make[2]: No se hace nada para `install-exec-am'.
make[2]: No se hace nada para `install-data-am'.
make[2]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2'
make[1]: Saliendo directorio `/home/matados2k/curso/supertux-0.1.2'
[root@fortaleza supertux-0.1.2]#
```

Y ¡tachan! ya tenemos supertux instalado en el ordenador. Simplemente ejecutando 'supertux' lo podremos usar.

Despedida.

Y después de esta pedazo de entrega, que mira que es larga, os espero hasta la próxima con la gestión básica de rpm.

Ningún CD de distribuciones de Linux fue maltratado en esta entrega y ningún Manolito tenia un tenedor.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 19. Instalando programas (II).

```
% make love
Make: Don't know how to make love. Stop.
```

Qué es eso de 'rpm'.

Las siglas significan RedHat Package Manager, y evidentemente lo crearon los de RedHat para su propia distribución, pero aparecieron otras distribuciones basadas en RedHat, como son por ejemplo Suse o Mandrake (actualmente Mandriva).

Como explicamos en la anterior entrega, es un formato binario que no necesita compilación aunque como veremos más adelante existen paquetes 'rpm' de fuentes que necesitan recompilarse (con extensión .src.rpm), que la única ventaja que trae con respecto a los fuentes de la semana pasada es que 'rpm' se encarga de compilar y crear un paquete 'rpm' binario ('rpm' es el gestor de paquetes y el nombre de la extensión del paquete, por si os estáis liando).

Cómo hacerme de paquetes 'rpm'.

Si tu distribución usa este sistema de paquetes, aparte de los que ya te vienen en tus cd's de la distribución, seguramente necesitarás instalar algo que no venga. El primer sitio donde buscar un 'rpm' para tu distribución es en la propia web del programa o la aplicación en cuestión. En el caso de que no lo den en formato 'rpm', o esos 'rpm' no son específicamente compilados para tu distribución y no quieres probar a ver si suena la campana (cuando uno se pone cabezón tiene que serlo hasta el final XD), siempre se puede buscar por la red de redes. Y ya puestos, y que os suelo dar todo mascadito, ahí van unas buenas direcciones donde buscar:

- http://rpmseek.com/
- http://rpm.pbone.net/
- http://rpmfind.rediris.es/
- http://rpmfind.net/

Y, sinceramente, si no encontráis lo que queréis en esas páginas no creo que lo encontréis en ninguna.

Cómo se interpreta normalmente el nombre de los paquetes.

Generalmente nos encontraremos paquetes nombrados más o menos como el de este ejemplo:

pepitoGrillo-3.0.0rc8-i586-rh9.rpm

La primera parte es el nombre, en este caso el nombre del supuesto programa es pepitoGrillo. A continuación tenemos la versión, en este caso tenemos que es 3.0.0rc8 (cuando os encontréis la coletilla 'rc', que significa 'release candidate', estáis delante de una versión 'candidata' a ser la definitiva por lo que pueden aparecer problemas, aunque generalmente suelen funcionar muy bien).

Seguidamente tenemos la arquitectura, en este caso i586 que se corresponde a un Pentium (esto no quiere decir que funcione sólo en Pentium, sino que funciona de ahí para arriba ya sea un P4 o un AMD XP). Hay muchas arquitecturas, así que solo haré una tabla con las más comunes que os encontraréis dentro de las arquitecturas de los PC.

```
i386 - 386
i486 - 486
i586 - Pentium
i686 - Pentium II
athlon - AMD Athlon
```

Con esto podéis deducir más o menos cómo van, y no es normal encontrar más de i686, siendo lo normal i386 e i586, tened en cuenta que los binarios se hacen para intentar llegar al máximo posible de gente.

Luego tenemos la distribución, en este caso 'rh9' que es RedHat 9. Cada distribución tiene sus siglas, así que pongo una tabla con algunas comunes:

```
rhX - RedHat X, siendo X el número de versión
fcX - Fedora Core X (RedHat)
mdkX - Mandrake
SuSe_X - Suse
```

Evidentemente hay muchas más, pero es para que os quedéis con la idea, lo que tenéis que enteraros es de las siglas de la vuestra.

Y por último la extensión, que puede ser o bien '.rpm', que es el formato binario, o bien '.src.rpm' y en ese caso nos encontraríamos ante fuentes.

Comencemos con el uso básico para 'rpm'.

El uso mas básico que le vamos a dar a los 'rpm' son sin duda instalar, desinstalar y actualizar. Para el ejemplo de hoy instalaremos el 'aMule', que es el clon de eMule más actual a día de escribir esta entrega. Para bajarnos lo necesario iremos a http://amule.sourceforge.net y nos iremos a su zona de descarga, en este caso no usaremos la versión estable del programa por estar totalmente desfasada y nos descargaremos (a día de la primera versión de este documento) la 2.0.0rc5 http://forum.amule.org/board.php?boardid=69&sid= , en mi caso me bajo la etiquetada como fc1 puesto que es la que uso, así que cada uno a la suya o la más parecida a la suya.

La orden básica para instalar es:

rpm -i Paquete.rpm

Ummm, parece fácil 'rpm' con la opción '-i', probemos:

```
[matados2k@fortaleza curso]$ rpm -i aMule-2.0.0-rc5_fc1.i686.rpm
error: Failed dependencies:
        wxGTK >= 2.4.2 is needed by aMule-2.0.0-rc5_fc1
        wxBase >= 2.4.2 is needed by aMule-2.0.0-rc5_fc1
        libcryptopp >= 5.1 is needed by aMule-2.0.0-rc5_fc1
[matados2k@fortaleza curso]$
```

Oh no!!! Dependencias, que no cunda el pánico, tendremos que buscar por nuestra cuenta los paquetes y para eso ya os he dicho cómo, antiguamente estaban en la propia página de aMule pero a día de hoy no están esos paquetes o yo no los he encontrado. También puede ser posible que hayan cambiado las dependencias en las nuevas versiones.

Una vez encontrado lo que nos falta, vamos a instalarlo todo:

```
[matados2k@fortaleza curso]$ rpm -i libcryptopp-5.2.1-1.i586-FC.rpm
error: cannot get exclusive lock on /var/lib/rpm/Packages
error: cannot open Packages index using db3 - Operación no permitida (1)
error: cannot open Packages database in /var/lib/rpm
[matados2k@fortaleza curso]$ su
Password:
[root@fortaleza curso]# rpm -i libcryptopp-5.2.1-1.i586-FC.rpm
[root@fortaleza curso]# rpm -i wxBase-2.4.2-1.i586.rpm
[root@fortaleza curso]# rpm -i wxGTK-2.4.2-1.i386.rpm
[root@fortaleza curso]# rpm -i aMule-2.0.0-rc5_fc1.i686.rpm
[root@fortaleza curso]# exit
exit
[matados2k@fortaleza curso]$
```

Como veis no se puede instalar sin ser root, así que ya sólo queda teclear 'amule' y ver cómo arranca nuestra aplicación.

Pues ahora damos un paso atrás.

Ahora lo que vamos a hacer es aprender a desinstalar, en este caso sólo desistalaremos aMule-2.0.0-rc5_fc1.i686.rpm para poner la 'rc4' y luego actualizar a 'rc5' para que veáis cómo se hace. Para desinstalar usamos:

rpm -e NombrePaquete

Y para actualizar usaremos la orden:

rpm -U Paquete.rpm

Veamos el ejemplo:

```
[matados2k@fortaleza curso]$ su
Password:
[root@fortaleza curso]# rpm -e aMule
[root@fortaleza curso]# rpm -i aMule-2.0.0rc4a-1.i686-FC.rpm
[root@fortaleza curso]# rpm -U aMule-2.0.0-rc5_fc1.i686.rpm
[root@fortaleza curso]# exit
exit
[matados2k@fortaleza curso]$
```

Observaréis que sólo es necesario usar el nombre de la aplicación para desinstalarla. Una buena idea es usar a la vez las opciones '-i' y '-U', ya que si el programa existe te lo actualiza y si no te lo instala :) .

Pero qué tengo y qué no tengo instalado.

Otra opción útil puede ser saber qué tienes o no instalado, para eso usamos:

rpm -qa

Y nos saldrá una lista seguramente enorme:

```
[matados2k@fortaleza curso]$ rpm -qa
tzdata-2003d-1
elfutils-libelf-0.89-2
....
mozilla-nss-1.4.1-18
samba-3.0.2-7.FC1
wxGTK-2.4.2-1
[matados2k@fortaleza curso]$
```

Así que, o bien usas este comando con la redirección a un fichero o una pipe y el comando 'more' para leer lo que salga tranquilamente, o vas a lo que buscas:

rpm -q NombrePaquete

Así que miramos nuestro aMule instalado:

```
[matados2k@fortaleza curso]$ rpm -q aMule
aMule-2.0.0-rc5_fc1
[matados2k@fortaleza curso]$
```

Como habéis podido ver se necesita root para instalar, desinstalar y actualizar, pero no para lo demás.

Otras opciones.

Otras opciones que os pueden interesar son '-F' para actualizar pero sólo si hay una versión más

antigua, '--force' para 'forzar' la instalación cuando sea necesario (como con algunos drivers de ATI :( ).

Despedida.

Bueno, terminamos esta entrega sin explicar cómo compilar los .src.rpm ya que no es el objetivo de esta entrega, y pasaremos en la próxima a ver los .deb. Un saludo y hasta la próxima.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 20. Instalando programas (III).

```
% rm Quayle-brains
rm: Quayle-brains nonexistent
```

Qué es eso de 'deb'.

Pues es el sistema de paquetes binarios usados por debian y todas las distribuciones que derivan de ella, que como ya comenté, no son pocas. Personalmente, después de usar rpm durante muchos años y ahora usar deb, me quedo con los deb.

¿Dónde hacerse de paquetes .deb?

Pues nada mejor que ir a la fuente de todo:

http://www.es.debian.org/distrib/packages#search_contents

La que podéis ver viene bastante completa, pudiendo buscar por arquitectura, por versiones (estable, de pruebas e inestable), incluso por cómo los tienen clasificados (sección), pero claro, son sólo los paquetes oficiales (que son muchísimos), personalmente no conozco paginas de búsqueda del estilo rmpfind.net para paquetes 'deb'. Y al igual que pasa con los 'rpm', en la propia web del programa suelen dar paquetes 'deb', aunque no siempre es así.

Pero además, los paquetes de debian están clasificados de la siguiente forma:

Main
Los paquetes de esta área son libres en sí mismos

Contrib
Los dueños del copyright de los paquetes de esta área les han dado licencia libre, pero dependen de otros programas que no son libres.

Non-Free (no libre)

Los paquetes de esta área tienen alguna condición de licencia onerosa que restringe su uso o redistribución.

Non-US/Main

Los paquetes de esta área son libres en sí mismos pero no pueden ser exportados desde un servidor en los Estados Unidos.

Non-US/Non-Free

Los paquetes de esta área tienen alguna condición de licencia onerosa que restringe su uso o redistribución. No pueden ser exportados de los Estados Unidos porque son paquetes de software de cifrado que no están gestionados por el procedimiento de control de exportación que se usa con los paquetes de Main o no pueden ser

almacenados en servidores en los Estados Unidos por estar sujetos a problemas de patentes.

Pero ojo, esta clasificación es sólo de los oficiales. Además, las dos últimas no se usan dado que no existen esas restricciones en Estados Unidos actualmente.

Cada distribución tiene su clasificación, como por ejemplo Ubuntu y derivadas donde los nombres son: Main, Restricted, Universe (Mantenidos por la comunidad) y Multiverse (igual que Non-Free).

Los 'deb' también tienen una nomenclatura más o menos estándar, como puede ser la del ejemplo:

aconnectgui_0.9.0rc2-1-7_i386.deb

Y como veis sigue una estructura similar a lo explicado para 'rpm' en la anterior entrega, por lo que me ahorro explicar de nuevo lo mismo ;).

Y sólo una cosa más antes de entrar al toro, también existen src.deb, que son paquetes que contienen el código fuente, pero al igual que en la anterior entrega tampoco los trataremos.

Aprendiendo el uso básico.

Si para 'rpm' el gestor de paquetes se llama igual, aquí la cosa cambia y el gestor es 'dpkg', del cual veremos lo básico para manejarnos.

Para el ejemplo de hoy vamos a instalar GetLeft que es un gestor de descargas de páginas webs, para ello nos dirigiremos a http://personal1.iddeo.es/andresgarci/getleft/english/ y concretamente a la zona de descarga http://personal1.iddeo.es/andresgarci/getleft/english/download.html de las cuales descargaremos para la entrega de hoy la versión 1.1.2 y la 1.1.1 en formato deb (faltaría menos XD).

La orden básica para instalar es:

dpkg -i Paquete.deb

Así que vamos a probar a instalar:

```
matados2k@fortaleza:~/curso$ su
Password:
fortaleza:/home/matados2k/curso# ls *.deb
getleft_1.1.1-2_all.deb  getleft_1.1.2-2_all.deb
fortaleza:/home/matados2k/curso# dpkg -i getleft_1.1.2-2_all.deb
Seleccionando el paquete getleft previamente no seleccionado.
(Leyendo la base de datos ...
141269 ficheros y directorios instalados actualmente.)
Desempaquetando getleft (de getleft_1.1.2-2_all.deb) ...
Configurando getleft (1.1.2-2) ...


fortaleza:/home/matados2k/curso#
```

Como veis, de momento es igual a 'rpm' , pero sólo de momento ;) . En este caso no han salido dependencias, en el caso de que salieran tendríais que resolverlas igual que las resolvimos con los 'rpm', pero en este caso sería diferente porque 'dpkg' sí que nos instala el paquete aunque no se cumplan las dependencias, cosa que también nos puede traer nuestros problemas (a la hora de mantener limpio el sistema si somos muy despistados, por ejemplo). De todas formas, y sin que sirva de precedente, si os pasa esto y no queréis liaros a buscar paquetes tendréis que tener fe en mí y creer que cuando ejecutéis 'apt-get -f instal' se solucionará todo, pero esto es algo que enseñaré en la próxima entrega.

Para ejecutarlo solamente teclead 'Getleft.tcl'. Si no os ejecuta seguro que es porque os falta el paquete 'curl', ya sabéis dónde buscarlo ;) .

Ahora nos arrepentimos y nos va a dar por desinstalarlo para poner la versión mas antigua, para ello vemos las formas que hay para desinstalar:

```
dpkg -r Nombre_Paquete
dkpg -P Nombre_Paquete
```

Vaya, esto se complica, ¿para qué dos formas de desinstalar?. Los paquetes 'deb' incluyen una configuración por paquete que optimiza cómo vamos a instalar el software que contiene, por decirlo de algún modo, y esto lo notaréis cuando instaléis mucho y el sistema os haga preguntas al instalar sobre cómo configurar el paquete.

Así que veamos las diferencias, con '-r' lo que hacemos es desinstalar el paquete pero sin desinstalar la configuración, y así si alguna vez reinstalamos no nos preguntara nada y usará la configuración existente. Esto se nos puede volver en nuestra contra en casos como, por ejemplo, que nos hayamos confundido en la configuración (en el caso de que nos la pida, que no es nuestro caso) y no funciona, si eliminamos y volvemos a instalar seguirá fallando.

Para eliminarlo todo optamos por '-P' que también elimina la configuración, así que manos a la obra:

```
fortaleza:/home/matados2k/curso# dpkg -P getleft
(Leyendo la base de datos ...
141491 ficheros y directorios instalados actualmente.)
Desinstalando getleft ...
Purgando ficheros de configuración de getleft ...
dpkg - atención: al desinstalar getleft, el directorio /usr/local/share'
no está vacío, no se borra.
dpkg - atención: al desinstalar getleft, el directorio /usr/local/lib'
no está vacío, no se borra.
dpkg - atención: al desinstalar getleft, el directorio /usr/local/doc'
no está vacío, no se borra.
dpkg - atención: al desinstalar getleft, el directorio /usr/local/bin'
no está vacío, no se borra.
dpkg - atención: al desinstalar getleft, el directorio /usr/local'
no está vacío, no se borra.
fortaleza:/home/matados2k/curso# dpkg -r getleft
dpkg - atención: el paquete getleft no está instalado.
 no se tendrá en cuenta la petición de desinstalarlo
fortaleza:/home/matados2k/curso#
```

Ya hemos desinstalado, y como podéis observar pasa como con los 'rpm', que sólo necesitamos indicarle el nombre del paquete sin más. Así que pasamos a instalar la versión más antigua:

```
fortaleza:/home/matados2k/curso# dpkg -i getleft_1.1.1-2_all.deb
Seleccionando el paquete getleft previamente no seleccionado.
(Leyendo la base de datos ...
141355 ficheros y directorios instalados actualmente.)
Desempaquetando getleft (de getleft_1.1.1-2_all.deb) ...
Configurando getleft (1.1.1-2) ...

fortaleza:/home/matados2k/curso#
```

Y esto lo hicimos solamente para actualizar la versión, que en este caso es simplemente con '-i' y como existe una versión antigua la actualizará.

```
fortaleza:/home/matados2k/curso# dpkg -i getleft_1.1.2-2_all.deb
(Leyendo la base de datos ...
141475 ficheros y directorios instalados actualmente.)
Preparando para reemplazar getleft 1.1.1-2 (usando getleft_1.1.2-2_all.deb) ...
Desempaquetando el reemplazo de getleft ...
Configurando getleft (1.1.2-2) ...

fortaleza:/home/matados2k/curso#
```

Fácil y cómodo, ¿verdad?, pero ahora nos queda aprender cómo saber si tenemos o no un paquete instalado:

dpkg -l [Nombre_paquete]

Si ponemos solo 'dpkg -l' nos listará todo lo instalado, cosa que puede no interesarnos, así que buscaremos lo que acabamos de instalar:

```
matados2k@fortaleza:~/curso$ dpkg -l getleft
Desired=Unknown/Install/Remove/Purge/Hold
| Estado=No/Instalado/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: mayúsc.=malo)
||/ Nombre         Versión        Descripción
+++-==============-==============-============================================
ii  getleft        1.1.2-2        Tcl/Tk WWW site grabber powered by curl
matados2k@fortaleza:~/curso$
```

Como veis nos da mucha información y nos las explica. ¿Alguien no la ve? (fijaos bien y convertid esos '|' en flechas mentalmente)

Y para terminar, ya que este es el uso más básico de 'dpkg', os dejo con una tabla de opciones, sacada directamente de 'man dpkg', que os puede resultar útil:

```
       dpkg -b | --build directorio [fichero]
           Construye un paquete Debian GNU/Linux.
       dpkg -c | --contents fichero
           Muestra el contenidos de un paquete Debian GNU/Linux.
       dpkg -e | --control fichero [directorio]
           Extrae la información de control de un paquete.
       dpkg -x | --extract fichero directorio
           Extrae los ficheros contenidos en el paquete.
       dpkg -f | --field  fichero [campo-control] ...
           Muestra el/los campo(s) de control de un paquete.
       dpkg --fsys-tarfile fichero
           Muestra el fichero tar contenido en el paquete Debian.
       dpkg -I | --info fichero [fichero-control]
           Muestra información sobre el paquete.
       dpkg -X | --vextract fichero directorio
           Extrae y muestra los nombres de ficheros contenidos en un paquete.
       dpkg -l | --list patrón-nombre-paquete ...
                  Lista los paquetes cuyo nombre encaja en el patrón dado.
        dpkg -s | --status nombre-paquete ...
                  Informa del estado del paquete especificado.
        dpkg -L | --listfiles paquete ...
                  Lista los ficheros instalados en el sistema, que pertenecen a
paquete.
         dpkg -S | --search patrón-búsqueda-ficheros ...
                  Busca un fichero en los paquetes instalados.
         dpkg -p | --print-avail paquete ...
                  Imprime información sobre el paquete, sacada de /var/lib/dpkg_
/available.
```

Despedida.

En la próxima entrega trataremos el uso básico de 'apt-get',y en las siguientes veremos algún gestor en modo gráfico.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 21. Instalando programas (IV).

In a world without barriers, who needs Gates and Windows?

Qué es eso de 'apt'.

Apt es un sistema de gestión de paquetes de software (utilizar este tipo de herramientas es la forma más indolora de instalar paquetes). Es originaria de Debian y está portada a distintas distribuciones no descendientes de ésta, como Red hat, Fedora, Mandrake/Mandriva, Suse y multitud de ellas. Así que éste será el único que veamos, ni urpmi ni yast ...

La propia página del 'man' indica que no hay aún interfaces amigables para este sistema ya que está aún en desarrollo, pero esto no es del todo cierto como veremos en la próxima entrega. Seguiremos vía línea de comandos para instalar.

La gran ventaja de apt, al igual que aplicaciones similares, es que tú le dices qué instalar y él solo te lo instala, además te resuelve e instala todas las dependencias, así que adiós a las instalaciones difíciles y bienvenido a instalaciones más sencillas, ordenadas y limpias que el siguiente, siguiente, siguiente de Windows. ¿No es una maravilla?

En cualquier caso, apt es un interfaz sencillo de 'dpkg' (o 'rpm' según el caso).

¿De dónde obtengo lo necesario para el conjunto de herramientas de apt?

Los que usen Debian o derivadas de ella como Knoppix, Linex, Guadalinex, Ubuntu y un puñado grande de ellas más, no tendrán problemas porque ya les viene de serie. Para los que no, ya pueden hacer búsquedas en www.google.es con las palabras apt-get y la distribución que quieres, aunque es posible que no lo encuentren para la suya.

¿Cómo, que no sabes cómo buscar? Me parece que si es así este curso no lo estás aprovechando bien.

Bueno, ya está instalado, pero no corras. Necesitamos saber algo más.

Bien, en este punto ya deberías tener instalado apt, pero aún nos quedan cosas antes de empezar. Apt en sí no es un comando, sino un conjunto de herramientas de las cuales veremos básicamente apt-get y apt-cache, que son las que normalmente cubrirán todas nuestras necesidades.

Apt, para manejar los paquetes, le da igual que sean deb o rpm, de ahí que esté portado a tantísimas distribuciones. Lo que apt necesita es que le indiques dónde tiene los repositorios de paquetes, los repositorios son los "lugares", por así decirlo, donde se encuentran los paquetes de los cuales queremos disponer, y puede ser cualquier fuente: cdrom, un ftp o vía http.

Nosotros de momento vamos a optar por instalar y actualizar con apt vía ftp o http, porque conseguiremos el software más actual ya que los repositorios suelen actualizarse frecuentemente. Para configurar esto debemos editar (como root) /etc/apt/sources.list y tendrá un aspecto como este:

```
#deb file:///cdrom/ sarge main


#deb cdrom:[Debian GNU/Linux testing _Sarge_ - Official Snapshot i386 Binary-3
(20040919)]/ unstable contrib main
#deb cdrom:[Debian GNU/Linux testing _Sarge_ - Official Snapshot i386 Binary-2
(20040919)]/ unstable contrib main
#deb cdrom:[Debian GNU/Linux testing _Sarge_ - Official Snapshot i386 Binary-1
(20040919)]/ unstable contrib main


#testing ETCH
deb ftp://ftp.fr.debian.org/debian/ testing main non-free contrib
#deb-src ftp://ftp.fr.debian.org/debian/ testing main non-free contrib


#inestable SID
#deb ftp://ftp.fr.debian.org/debian/ unstable main non-free contrib
#deb-src ftp://ftp.fr.debian.org/debian/ unstable main non-free contrib
```

Este es un sources.list ya configurado para una Debian Etch, y puede que el vuestro esté ya configurado o no, pero nos servirá para explicar qué significa cada cosa.

1º) Todo lo que se encuentre después de # se considera un comentario y será ignorado, como podéis ver yo tengo comentadas muchas líneas.

2º) El formato de las líneas es el siguiente:

tipo_de_paquete fuente distribución [componente1] [componente2] [...]

En "tipo de paquete" va o bien 'deb', 'deb-src' o bien 'rpm', 'rpm-src' dependiendo del tipo de paquetes que usemos. En "fuente" indicamos dónde está el repositorio, que suele ser o bien algo como:

cdrom:[Debian GNU/Linux testing _Sarge_ - Official Snapshot i386 Binary-3 (20040919)]/ lo que nos indica que es un cd y en este caso concreto el de instalación.

O bien es una dirección http o ftp, como podéis ver en el ejemplo. Lo siguiente es la distribución, en mi caso al usar Debian aparecen unstable y testing, que son 2 de las 3 ramas que tiene Debian (stable, testing y unstable con los nombres de Sarge, Etch y Sid respectivamente). Y por último, los componentes que pueden existir o no, en mi caso se corresponden con la forma de organizar los paquetes de Debian. En cada distribución estas cosas cambian, depende de cómo se estructuren internamente.

Pero tranquilos, que aunque como veis es fácil, no tendréis que crear las líneas vosotros mismos (pero os lo explico porque es bueno que sepáis qué significan), ni nada parecido a investigar cómo es internamente vuestra distribución. Nosotros sólo nos limitaremos a buscar por internet cuáles son las líneas que tenemos que añadir. ¿A que es fácil?

Así que sólo os queda buscar esas líneas, preferiblemente cogiendo siempre de las fuentes oficiales. Tranquilos, es muy fácil y la red está llena de ellas ;) y dependiendo de qué programas o aplicaciones necesitéis, necesitaréis más o menos repositorios.

Sin duda alguna, en esto de repositorios y número de paquetes gana por goleada Debian, que es una de las grandes virtudes de esta distribución, aunque no sea precisamente la más fácil de instalar y poner en marcha. Sabiendo buscar bien repositorios es posible que nunca necesitéis compilar nada para esta distribución. Mi experiencia con otras distribuciones como Fedora es que sus repositorios y paquetes disponibles en ellos son realmente pobres.

Ya lo tengo todo, que empiece la fiesta.

Ya hemos llegado a lo interesante, y no me digáis que os ha sido difícil llegar hasta aquí porque no me lo creo. Para empezar con nuestro sources.list ya configurado, lo que tenemos que hacer es sincronizar el índice de paquetes respecto a las fuentes (los repositorios apuntados por nuestro sources.list):

apt-get update

Nada más fácil que ejecutar esto cada vez que cambiemos las fuentes del sources.list. Como los repositorios suelen actualizarse muy a menudo, sobre todo los de Debian, es conveniente que ejecutéis esto con frecuencia para no encontrar problemas a la hora de instalar. Veamos un ejemplo:

```
matados2k@fortaleza:~$ su
Password:
fortaleza:/home/matados2k# apt-get update
Obj http://non-us.debian.org testing/non-US/main Packages
Des:1 http://debian.tu-bs.de unstable/main Packages [17,6kB]
Obj http://non-us.debian.org testing/non-US/main Release
Obj http://non-us.debian.org testing/non-US/contrib Packages
Obj http://non-us.debian.org testing/non-US/contrib Release
Obj http://non-us.debian.org testing/non-US/non-free Packages
Obj http://non-us.debian.org testing/non-US/non-free Release
Des:2 http://debian.tu-bs.de unstable/main Release [112B]
Des:3 ftp://ftp.fr.debian.org testing/main Packages [3151kB]
Des:4 ftp://ftp.fr.debian.org testing/main Release [81B]
Des:5 ftp://ftp.fr.debian.org testing/non-free Packages [58,6kB]
Des:6 ftp://ftp.fr.debian.org testing/non-free Release [85B]
Des:7 ftp://ftp.fr.debian.org testing/contrib Packages [55,8kB]
Des:8 ftp://ftp.fr.debian.org testing/contrib Release [84B]
Descargados 3284kB en 3m32s (15,5kB/s)
Leyendo lista de paquetes... Hecho
fortaleza:/home/matados2k#
```

Veis qué fácil, no tenéis que hacer nada por vuestra parte, sólo esperar.

Siempre que hagáis un 'apt-get update' es bueno hacer:

apt-get check

que no es más que una herramienta de diagnóstico. Actualiza la caché de paquetes , vuelve a crear un nuevo árbol de dependencias y busca dependencias imposibles de resolver, aunque si no entendéis nada de esto no importa, simplemente sabed que es bueno realizarlo, sobre todo para saber si todo ha salido bien:

```
fortaleza:/home/matados2k# apt-get check
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
fortaleza:/home/matados2k#
```

Instalar y actualizar, 2 en 1.

Apt, si está un paquete ya instalado, lo actualiza a la última versión, y en caso contrario lo instala. Lo mismo con todas sus dependencias, que se resuelven automáticamente Qué más podemos pedir cuando es especial para cómodos :) :

```
apt-get install lista_de_paquetes
```

Para ver un ejemplo de esto vamos a instalar hoy un pequeño editor de texto compatible en gran medida con los '.doc' de Windows, Abiword (aunque si queréis un buen paquete ofimático ya sabéis, OpenOffice.org):

```
fortaleza:/home/matados2k# apt-get install abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se instalarán los siguientes paquetes extras:
  gnome-core gnome-desktop-environment
Paquetes sugeridos:
  gnome-audio
Paquetes recomendados
  dasher gnome-mag gnopernicus gok gnome-accessibility-themes
Los siguientes paquetes se ELIMINARÁN:
  abiword-gnome gnome gnome-office
Se instalarán los siguientes paquetes NUEVOS:
  abiword
Se actualizarán los siguientes paquetes:
  gnome-core gnome-desktop-environment
2 actualizados, 1 se instalarán, 3 para eliminar y 22 no actualizados.
Necesito descargar 1905kB de archivos.
Se liberarán 213kB después de desempaquetar.
¿Desea continuar? [S/n] s
Des:1 ftp://ftp.fr.debian.org testing/main abiword 2.0.14-1 [1885kB]
Des:2 ftp://ftp.fr.debian.org testing/main gnome-desktop-environment 61 [9960B]
Des:3 ftp://ftp.fr.debian.org testing/main gnome-core 61 [9782B]
Descargados 1905kB en 1m28s (21,4kB/s)
(Leyendo la base de datos ...
...
...
Desempaquetando el reemplazo de gnome-core ...
Configurando abiword (2.0.14-1) ...
Configurando gnome-core (61) ...
Configurando gnome-desktop-environment (61) ...
fortaleza:/home/matados2k#
```

Podéis comprobar que 'apt-get' con su opción 'install' se ha ocupado de todo, incluso de eliminarme paquetes que ya no serán necesarios, cosa que no comenté antes. Como veis es lo mejor para mantener limpio el sistema.

Desinstalar tiene también 2 formas.

Desinstalar tiene 2 formas, aunque sólo para los que usan paquetes 'deb', y para eso no hay más que acordarse de la entrega pasada de que los paquetes 'deb' llevan también archivos de configuración de paquetes.

apt-get remove lista_de_paquetes
apt-get --purge remove lista_de_paquetes

El primero es el que elimina el paquete que le indicas, y el segundo el que además elimina los ficheros de configuración de paquete que nos serán necesarios cuando instalemos de forma errónea y desinstalemos e instalemos de nuevo. Para ello, como siempre, un ejemplo:

```
fortaleza:/home/matados2k# apt-get remove abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Los siguientes paquetes se ELIMINARÁN:
  abiword abiword-common
0 actualizados, 0 se instalarán, 2 para eliminar y 22 no actualizados.
Necesito descargar 0B de archivos.
Se liberarán 11,0MB después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ...
146960 ficheros y directorios instalados actualmente.)
Desinstalando abiword-common ...
Desinstalando abiword ...
fortaleza:/home/matados2k# apt-get --purge remove abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
El paquete abiword no esta instalado, no se eliminará
0 actualizados, 0 se instalarán, 0 para eliminar y 22 no actualizados.
fortaleza:/home/matados2k# apt-get install abiword
.....
fortaleza:/home/matados2k# apt-get --purge remove abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Los siguientes paquetes se ELIMINARÁN:
  abiword* abiword-common*
0 actualizados, 0 se instalarán, 2 para eliminar y 22 no actualizados.
Necesito descargar 0B de archivos.
Se liberarán 11,0MB después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ...
146960 ficheros y directorios instalados actualmente.)
Desinstalando abiword-common ...
Desinstalando abiword ...
fortaleza:/home/matados2k#
```

Como veis 'apt-get remove' también elimina todo lo que ya no necesitemos, incluidas las dependencias, que ya no son usadas por ningún otro paquete.

¡Oh no! Tenemos un problema.

Muchas veces si no hacemos 'apt-get update' de forma habitual nos podemos encontrar con el problema de que no pueden instalarse los paquetes porque ya no se encuentran, o bien has instalado algo que no venía en los repositorios y trastes de 'dkpg' y ocurrieron dependencias, en cualquier caso siempre que se encuentren dependencias rotas. Como podréis suponer si seguís el curso este problema es más probable que se dé con los que usan 'deb' que con los que usan 'rmp', ya que 'dpkg' instala aunque encuentre dependencias y a 'rpm' en cambio hay que decirle que fuerce la instalación ya que si encuentra dependencias por defecto no instalaría.

Pero gracias a Dios este es un problema de lo más sencillo de solucionar. En el caso de que sea por no sincronizar con 'apt-get update', realizamos este paso. En cualquier otro caso, realizar:

apt-get -f install

Lo que hace es intentar arreglar un sistema con dependencias actualmente rotas, y generalmente con un buen fichero 'sources.list' nuestro problema desaparecerá. En un sistema sin dependencias rotas como el mío, la ejecución de esta orden daría el siguiente resultado:

```
fortaleza:/home/matados2k# apt-get -f install
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
0 actualizados, 0 se instalarán, 0 para eliminar y 22 no actualizados.
fortaleza:/home/matados2k#
```

Cortamos aquí, despedida y cierre.

Como esta entrega se me ha hecho increíblemente larga en comparación con el resto, he tenido que cortar y hacerla en dos. Ambas os las entrego al mismo tiempo para no quedaros a medias.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 22. Instalando programas (V).

```
% "How would you rate Bush's incompetence?
Unmatched ".
```

Ponte a la última y actualiza tu sistema completo, sé más chulo que un 8 verde pistacho ;) .

Si lees bien, con apt es posible actualizar el sistema completo e incluso la distribución completa a la última versión, con lo que con un buen sistema administrado sólo necesitarás una instalación.

Y ahora que lo pienso, ¿cómo actualizas un sistema Windows?, ah sí, haces copia de seguridad de tus datos, casi seguro que te conviene formatear tu partición e incluso dependiendo del caso cambiar el formato de sistema de archivos y luego empiezas a instalar el nuevo Windows (en la mayoría de los casos pirata XD), y luego a instalar de nuevo los programas desde 0. Si está claro... con Windows son todo facilidades XD sobre todo para mantener una limpieza, un orden y sobre todo para pasar por caja XD. Bill, tontorrón, no te enfades que Windows tiene muchos.. esto... colorines.

Hay dos formas de actualizar, cada una de ellas con sus peculiaridades, y ambas dos son absurdas si antes no se realiza un 'apt-get update'. Veamos cuáles son y qué matices tiene cada una de ellas:

```
apt-get upgrade
apt-get dist-upgrade
```

La primera se puede decir que es la más respetuosa, ya que se usa para instalar la versión más nueva de todos los paquetes instalados en el sistema provenientes de alguna de las fuentes listadas el 'sources.list'. Los paquetes instalados con una nueva versión disponible son descargados y actualizados, bajo ninguna circunstancia se desinstalarán paquetes, o se instalarán paquetes nuevos. Las nuevas versiones de programas instalados que no puedan ser actualizados sin cambiar el estado de instalación de otros paquetes no se instalarán, manteniéndose la versión actual.

Y la segunda, además de realizar las acciones de la primera, maneja inteligentemente los cambios de dependencias debidos a nuevas versiones de paquetes, apt-get tiene un sofisticado sistema de resolución de conflictos, si es necesario tratará de actualizar los paquetes más importantes a costa de los menos importantes.

Normalmente, si se hace la segunda no se necesitará realizar la primera, pero si se hace la primera entonces la segunda hará aún más actualizaciones. Veamos un ejemplo de ambas:

```
fortaleza:/home/matados2k# apt-get upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se actualizarán los siguientes paquetes:
  docbook-dsssl exim4 exim4-base exim4-config exim4-daemon-light fwbuilder
  fwbuilder-common fwbuilder-linux gallery gftp gftp-common gftp-gtk gftp-text
  groff-base klogd lha libfwbuilder6 libmysqlclient12 libsensors3 libssl0.9.6
  libtag1 libxine1 manpages menu mysql-client mysql-common mysql-server
  sysklogd tasksel xprt-common xprt-xprintorg zlib1g zlib1g-dev
33 actualizados, 0 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 23,9MB de archivos.
Se utilizarán 408kB de espacio de disco adicional después de desempaquetar.
¿Desea continuar? [S/n] s
Des:1 ftp://ftp.fr.debian.org testing/main zlib1g-dev 1:1.2.2-3 [488kB]
....
Des:33 ftp://ftp.fr.debian.org testing/main libssl0.9.6 0.9.6m-1 [1755kB]
Descargados 23,9MB en 18m31s (21,5kB/s)
Preconfiguring packages ...
....
Configurando exim4-base (4.34-7) ...
....
fortaleza:/home/matados2k# apt-get dist-upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Calculando la actualización... Listo
0 actualizados, 0 se instalarán, 0 para eliminar y 0 no actualizados.
fortaleza:/home/matados2k#
```

En este caso 'apt-get dist-upgrade' no hizo nada porque no fue necesario, ya que 'apt-get upgrade' realizó todo lo imprescindible.

Para actualizar tu distribución lo normal es cambiar los repositorios de la que usas por la que deseas, por ejemplo, para pasar de Debian Etch a Debian Sid cambiamos en sources.list las fuentes de una por la de la otra y nos lanzamos con 'apt-get update','apt-getcheck' y 'apt-get dist-upgrade'. Aunque esto funciona bien en Debian, ojo, puede que en otras distribuciones no sea así y podáis tener problemas.

Limpia la caché del apt.

Apt lo que hace es bajar los paquetes y luego hacer lo que sea necesario con ellos. Para ello usa un directorio como caché situado en /var/cache/apt/archives y éste nunca se vacía, a menos que se haga manualmente borrando los archivos, cosa que no recomiendo (aunque a mí nunca me dio problemas) o bien mediante otras dos opciones del apt con sus pequeñas diferencias.

Si no lo borras puede llegar a convertirse con el paso del tiempo en un problema de espacio, por

ejemplo, yo que suelo ser descuidado b tengo al tamaño de 1.4 GB, aunque es conveniente también no tenerla siempre vacía, sobre todo cuando instalas y vuelves a instalar un mismo paquete, ya que lo que encuentre en la caché no lo bajará.

Las dos formas son:

```
apt-get clean
apt-get autoclean
```

La primera borra totalmente el repositorio local que contiene los ficheros descargados, y la segunda sólo borrará aquellos paquetes que ya no pueden ser descargados, o son claramente inservibles. Esto permite mantener la caché durante largos periodos de tiempo sin que aumente su tamaño sin control.

Vamos a ver un ejemplo de ambas, aunque esta vez empezaremos por la segunda:

```
fortaleza:/home/matados2k# apt-get autoclean
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Del a2ps 1:4.13b-4 [715kB]
Del alsa-base 1.0.5a-3 [47,5kB]
...
Del gftp 2.0.17-6 [34,6kB]
Del manpages-dev 1.67-2 [1044kB]
Del mysql-server 4.0.21-6 [0B]
fortaleza:/home/matados2k#
```

Con esto he pasado de una caché de 1.4GB a una de 776.7 MB.

```
fortaleza:/home/matados2k# apt-get clean
fortaleza:/home/matados2k#
```

Y con esto la caché esta vacía.

Yo sólo quiero que se descarguen los paquetes.

Puede que nos interese que al instalar o actualizar sólo descargue los paquetes a la caché y no haga nada más con ellos. Para eso usamos la opción '-d' quedando las ordenes así:

```
apt-get -d install lista_de_paquetes
apt-get -d upgrade
apt-get -d dist-upgrade
```

Esto puede ser muy útil si vas a instalar y no vas a estar presente como en el caso de los 'deb', que muchas veces lanzan preguntas sobre cómo se tienen que configurar los paquetes, o también si vamos a instalar algo y como tenemos una conexión lenta queremos copiar los paquetes de la caché a un directorio para no volverlos a bajar nunca más :)

Veamos el ejemplo sólo con la instalación:

```
fortaleza:/home/matados2k# apt-get -d install abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Se instalarán los siguientes paquetes extras:
  abiword-common
Paquetes sugeridos:
  abiword-plugins abiword-plugins-gnome abiword-doc
Paquetes recomendados
  abiword-help
Se instalarán los siguientes paquetes NUEVOS:
  abiword abiword-common
0 actualizados, 2 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 3371kB de archivos.
Se utilizarán 11,0MB de espacio de disco adicional después de desempaquetar.
¿Desea continuar? [S/n] s
Des:1 ftp://ftp.fr.debian.org testing/main abiword-common 2.0.14-1 [1486kB]
Des:2 ftp://ftp.fr.debian.org testing/main abiword 2.0.14-1 [1885kB]
Descargados 3371kB en 2m42s (20,7kB/s)
Descarga completa y en modo de sólo descarga
fortaleza:/home/matados2k# apt-get install abiword
Leyendo lista de paquetes... Hecho
...
Se utilizarán 11,0MB de espacio de disco adicional después de desempaquetar.
¿Desea continuar? [S/n] s
Seleccionando el paquete abiword-common previamente no seleccionado.
(Leyendo la base de datos ...
...
Configurando abiword-common (2.0.14-1) ...


fortaleza:/home/matados2k#
```

Podéis observar cómo simplemente se lo baja y para. Al ejecutar la misma orden sin '-d', como los encuentra en caché, ya no baja nada y en este caso se instala.

Pero si quiero reinstalar algo ya instalado..

Imaginemos que se nos ha corrompido el ejecutable o ficheros de algo que ya tenemos instalado, nuestra primera ocurrencia sería:

```
fortaleza:/home/matados2k# apt-get install abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
abiword ya está en su versión más reciente.
0 actualizados, 0 se instalarán, 0 para eliminar y 0 no actualizados.
fortaleza:/home/matados2k#
```

Uhmm... realmente un problema, ya que por defecto si está instalado ya no reinstala encima, pero esto lo podemos solucionar fácilmente con la opción '--reinstall'. Veamos el ejemplo:

```
fortaleza:/home/matados2k# apt-get --reinstall install abiword
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
0 actualizados, 0 se instalarán, 1 reinstalados, 0 para eliminar y 0 no
actualizados.
Se necesita descargar 0B/1885kB de archivos.
Se utilizarán 0B de espacio de disco adicional después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ...
146924 ficheros y directorios instalados actualmente.)
Preparando para reemplazar abiword 2.0.14-1 (usando .../abiword_2.0.14-
1_i386.deb) ...
Desempaquetando el reemplazo de abiword ...
Configurando abiword (2.0.14-1) ...
fortaleza:/home/matados2k#
```

Como veis es fácil ganar el pulso si se conocen las herramientas necesarias.

Ya ya, todo está bien pero falta algo, ¿Y si no conozco el nombre de lo que quiero instalar?

Pues también tenemos un nuevo comando dentro de las utilidades de apt del cual sólo veremos una opción, y es la siguiente:

apt-cache search lista_de_palabras

Simplemente realiza una búsqueda de la expresión regular dada en todo el texto de todos los paquetes disponibles. Busca la expresión regular tanto en el nombre de los paquetes como en su descripción, y muestra el nombre del paquete y una pequeña descripción de éste. Así que vamos sin más a buscar juegos que sean como el Bubble:

```
fortaleza:/home/matados2k# apt-cache search bubble
bubblefishymon - system load dockapp with a duck
bubblemon - Bubbling Load Monitoring GNOME Applet
emacs-goodies-el - Miscellaneous add-ons for Emacs
fb-music-high - High quality, large music files for Frozen-Bubble
fb-music-low - Lower quality, small music files for Frozen-Bubble
frozen-bubble - Pop out the bubbles !
frozen-bubble-data - Data files for Frozen-Bubble
gdesklets-data - displays and sensors for gdesklets
gkrellm-bfm - system load plugin for gkrellm with a duck
junior-puzzle - Debian Jr. Puzzles
wmbubble - A system-load meter for Window Maker that features a duck
wmfishtime - Dockable clock app for WMaker, BlackBox, E, SawFish etc
xbubble - A nice Puzzle Bubble clone
xbubble-data - Data files for XBubble, a nice Puzzle Bubble clone
fortaleza:/home/matados2k#
```

¿Hace falta a estas alturas deciros cómo instalar alguno de la lista?

No me funcionan cosas de las que explicas.

Es posible, ya que si usas un port del apt (originario de Debian) puedes encontrartecon el problema de que hay opciones inexistentes por ser inútiles para el tipo de paquetes que vas a manejar, o que aún esa parte no está implementada

En cualquier caso siempre hay que consultar 'man apt' , 'man apt-get' y 'man apt-cache' de tu sistema, que aparte de ver las diferencias verás las múltiples opciones que existen que yo no explico, y algunos comandos más dentro de apt que tampoco hemos aprendido.

En cualquier caso creo que con lo aquí visto tenéis la mayor parte de vuestras necesidades cubiertas, y los cambios si es que existen son mínimos.

Despedida y cierre.

Espero que estas dos últimas entregas te sean de utilidad y sean tan digeribles como intento que sean todas mis entregas. En la próxima veremos ya un par de herramientas gráficas para la instalación de paquetes y daremos por terminada esta serie de entregas. Un saludo y hasta la próxima.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 23. Instalando programas (VI).

```
% cat catfood
cat: cannot open catfood
```

Por fin, instaladores gráficos .

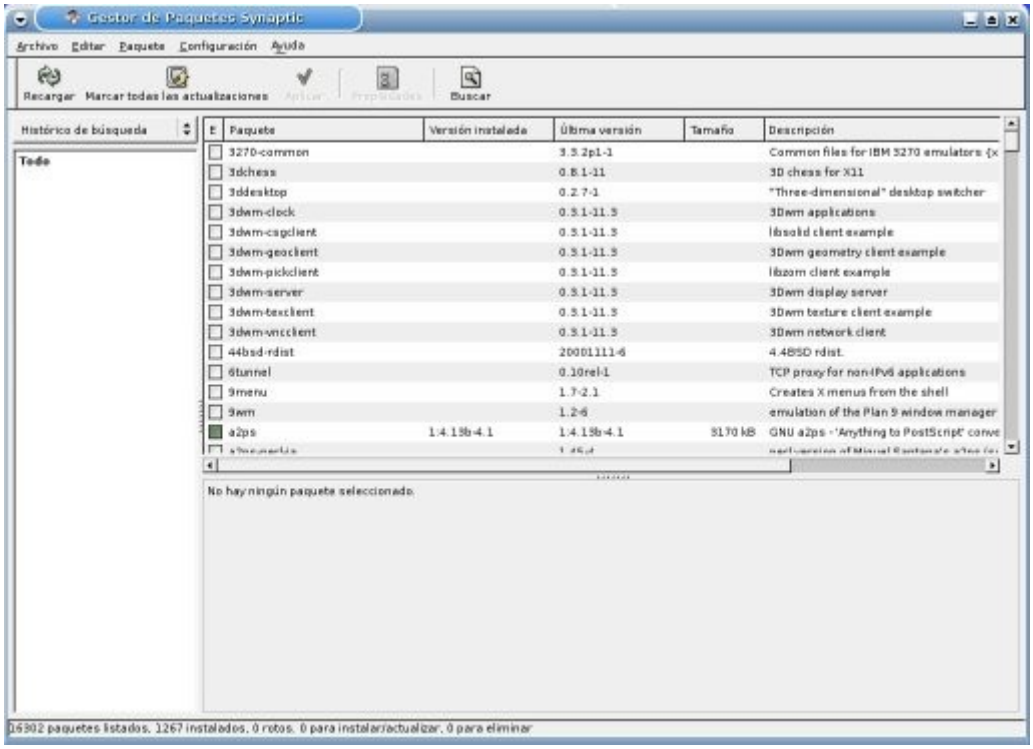
Como ya comenté vamos a ir viendo ya aplicaciones gráficas para instalar los paquetes, de entre ellas hay que destacar dos, una perteneciente a los escritorios KDE como es Kpackage y otro (que es el que veremos) llamado Synaptic, aunque ambos nos ofrecen los mismo. Seguro que en tu distribución hay un paquete para Synaptic así que si no lo tienes instalado es hora de hacerlo.

La forma en la que vamos a ver la aplicación es sencilla, ya que Synaptic es un front-end gráfico de apt, como ya sabemos el uso de apt no vamos a explicar de nuevo el funcionamiento, sino ver cómo haríamos gráficamente lo que aprendimos en las dos entregas anteriores

Podéis ver su pagina Web http://www.nongnu.org/synaptic/index.html para más información.

Arrancando la aplicación.

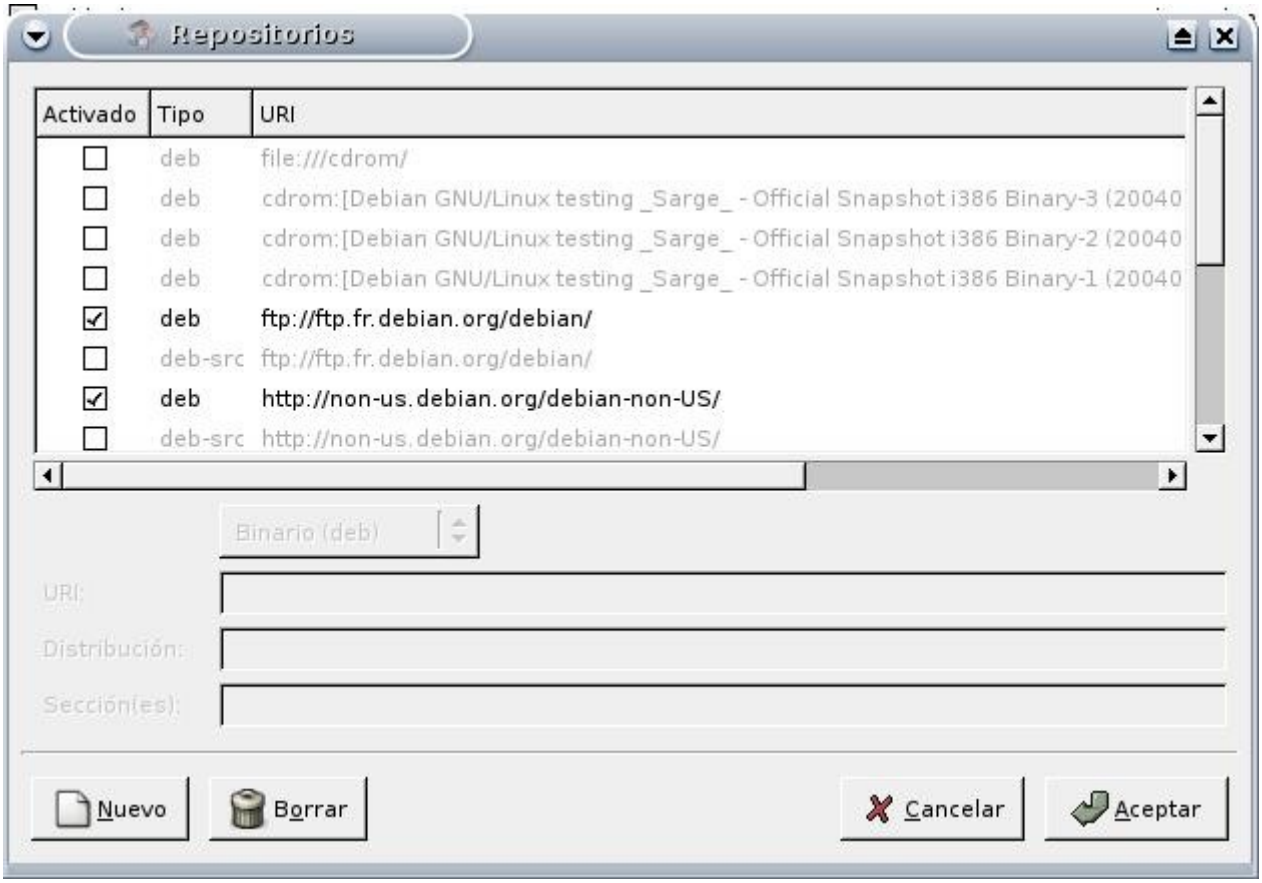
Si no encontráis en el menú de vuestro escritorio favorito Synaptic (cosa rara, buscad bien) podéis arrancarlo desde una consola (siendo root) con el comando 'synaptic'. Si arrancáis desde el menú y suponiendo, como debería ser, que estáis desde un usuario normal os pedirá la clave de root, la introducimos y tendremos algo parecido a esto:



Manejando las fuentes

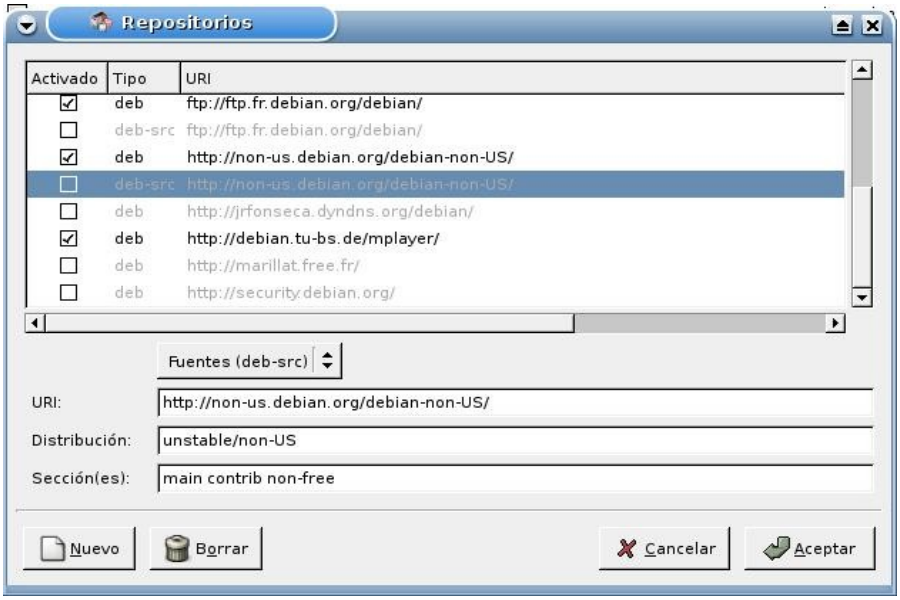
Lo primero que vamos a aprender es a manejar las fuentes (acordaos de /etc/apt/sources.list que es lo

que realmente vamos a tocar pero de forma gráfica), nos dirigimos al menú 'Configuración' y dentro de éste elegimos 'Repositorios', con lo que obtenemos la siguiente ventana:



Si recordamos cuando lo hacíamos manualmente, para comentar una línea y ésta no tenga efecto usábamos un '#'. Ahora simplemente desmarcamos una línea y listo, con lo que las fuentes efectivas son las que tenemos en negrita.

Para modificar una línea la seleccionamos y podremos editar su contenido en la parte inferior:

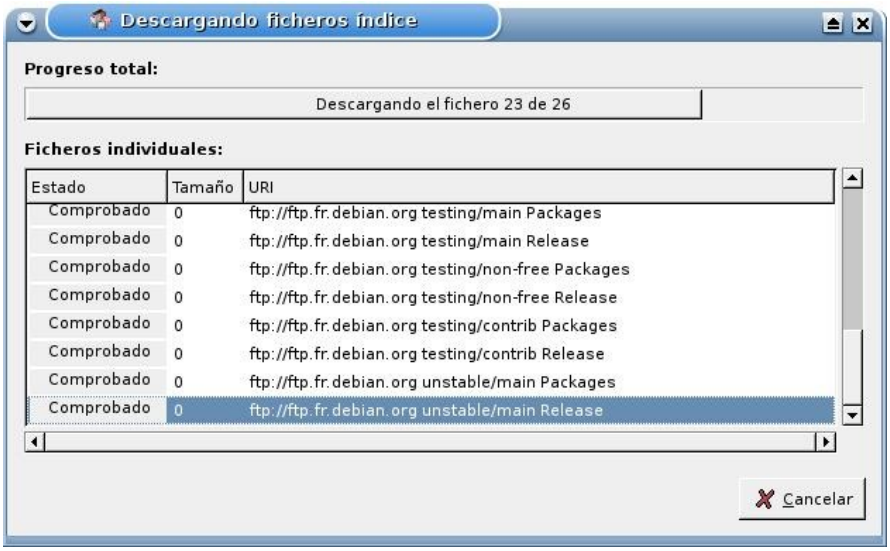


Para eliminarlas simplemente seleccionas la que quieres y pulsamos sobre 'borrar', así que si quieres

probar puedes pulsar sobre 'Nuevo' para crear una nueva entrada y probar a editarla y luego borrarla.

Ya tengo a mi gusto las fuentes ¿Y ahora, qué?

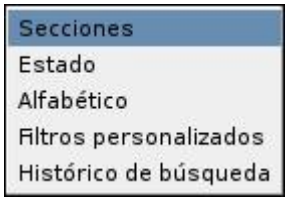
Una vez terminado el paso anterior tendremos que hacer el equivalente a 'apt-get update' para sincronizar el índice de paquetes respecto a las fuentes, así que sin más pulsamos en el botón 'Recargar' de la pantalla principal del programa, con lo que se nos abre un cuadro de diálogo de progreso tal que así:



Instalar, desinstalar, reinstalar y actualizar. Todo al alcance de un simple click.

Lo primero es saber qué queremos instalar, porque en la pantalla principal aparecen todos los paquetes disponibles y la verdad, así no hay quien se aclare (imaginaos mi caso con una lista de 22.206 paquetes entre Debian Etch y Debian Sarge). Lo primero que vamos a hacer es que se ordenen de alguna forma más lógica.

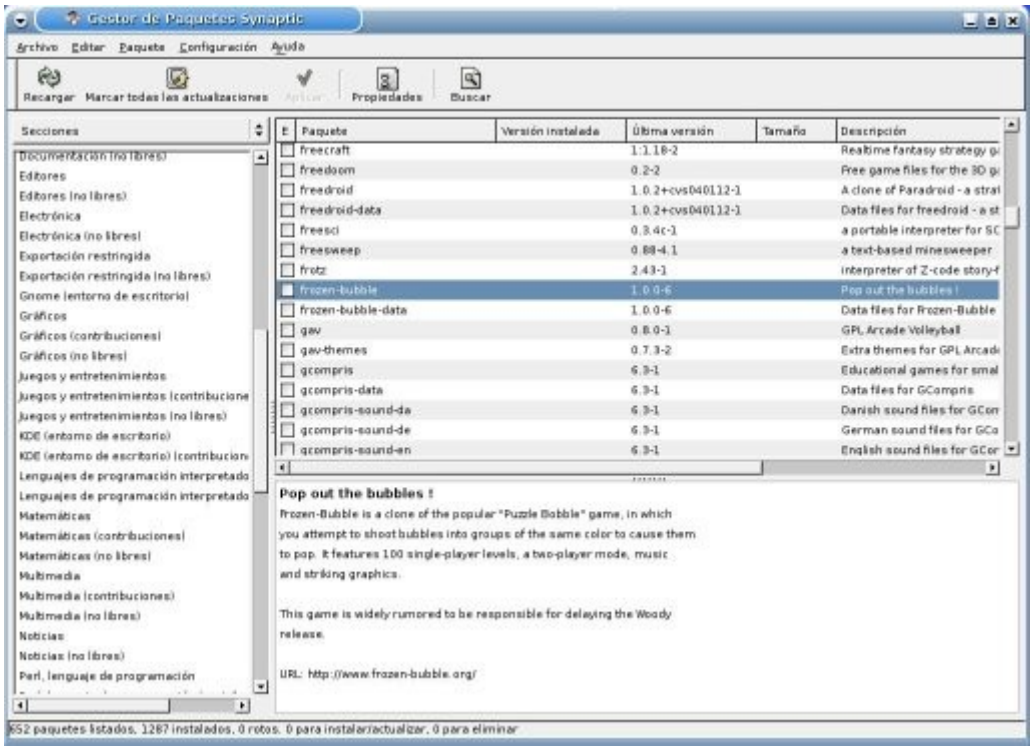
Para empezar, nos fijamos arriba a la izquierda donde pone “Histórico de búsqueda” (o cualquier otra cosa del siguiente recuadro) y pulsamos:



Vemos las posibilidades de ordenación que se nos dan, de las cuales la más interesante a mi modo de ver es por Secciones y quizás por orden Alfabético, la cosa es que investigues por tu cuenta, de momento para la explicación elegiremos Secciones.

Estupendo, así sí que da gusto, vemos en la parte izquierda muchas secciones muy ordenadas por temas, así que vamos a picar por ejemplo en... juegos y entretenimiento (según el sistema que tengáis tendréis uno u otros o unos muy parecidos). Ahora vamos a buscar el frozen-bubble en la lista superior derecha y lo seleccionamos (no marcarlo) y vemos que nos aparece en la parte inferior izquierda la descripción del paquete, estupendo :) no sólo buscamos visualmente sino que además

podemos saber de qué se trata.

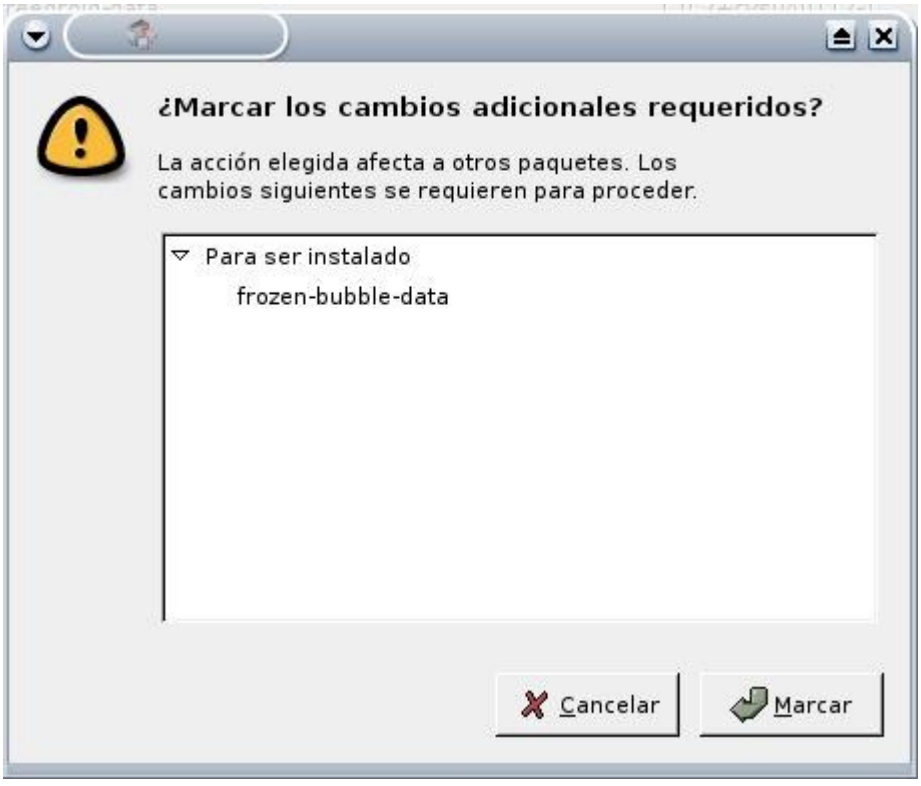


En la columna 'E' podéis ver un cuadradito que nos indica en qué estado están estos paquetes (instalado, sin instalar ...), los posibles estados los podemos ver en el menú ayuda y son los siguientes:

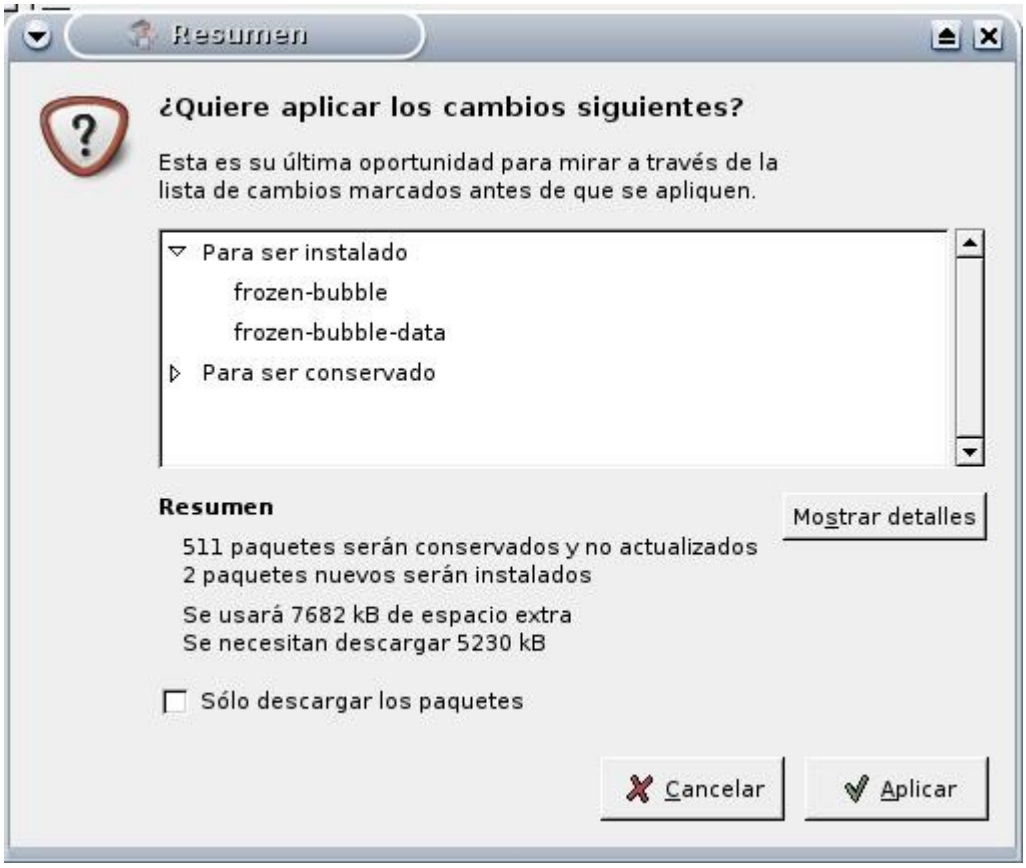
Pues bien, nos lo ponen todo en bandeja, sabiendo esto sólo tenemos que marcar el paquete con lo que queremos hacer seleccionándolo y pulsando el botón secundario del ratón:

Estupendo, tenemos un todo en 1, desde aquí según el estado del paquete aparecerán activas las opciones posibles a realizar, desmarcar (si estaba marcada) para no hacer nada, reinstalar un paquete, actualizarlo, eliminarlo y eliminarlo completamente(acordaos de que los paquetes 'deb' contenían ficheros de configuración que sólo se eliminaban con la opción --purge del 'apt-get').

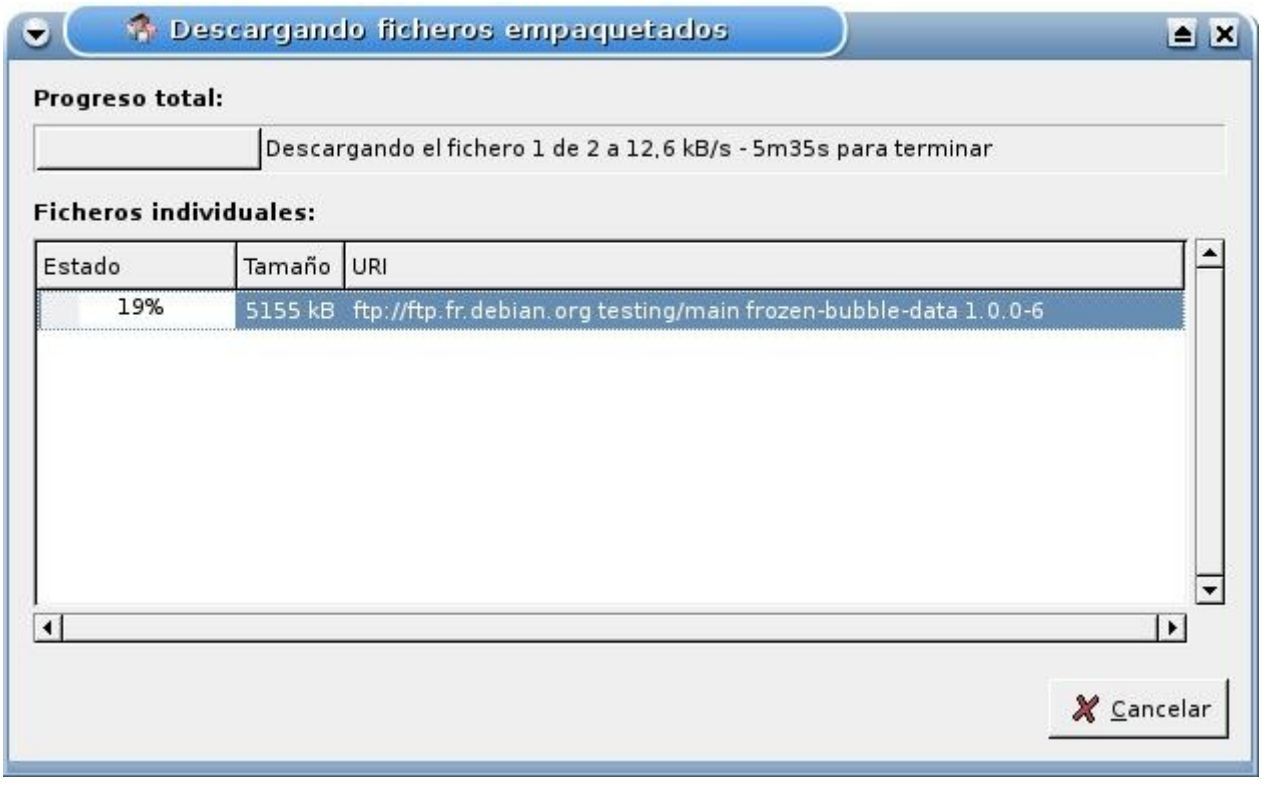
Pues ya está, lo marco para instalar y automáticamente se me marcarán todas sus dependencias para instalarse :)



Ya sabemos cómo realizar todas las órdenes, pero pulsamos y no lo hace, sino que 'lo marca'. Esto nos da la posibilidad de indicar todo tipo de acciones para luego pulsar en 'Aplicar' y ver cómo nuestros deseos se hacen realidad:



Vemos la información de lo que se realizará y nos da la opción de 'Sólo descargar los paquetes' como el '-d' de 'apt-get', 1,2,3.. YA



Y después de comprobarlo malo que es tener un P2P activo si eres impaciente:

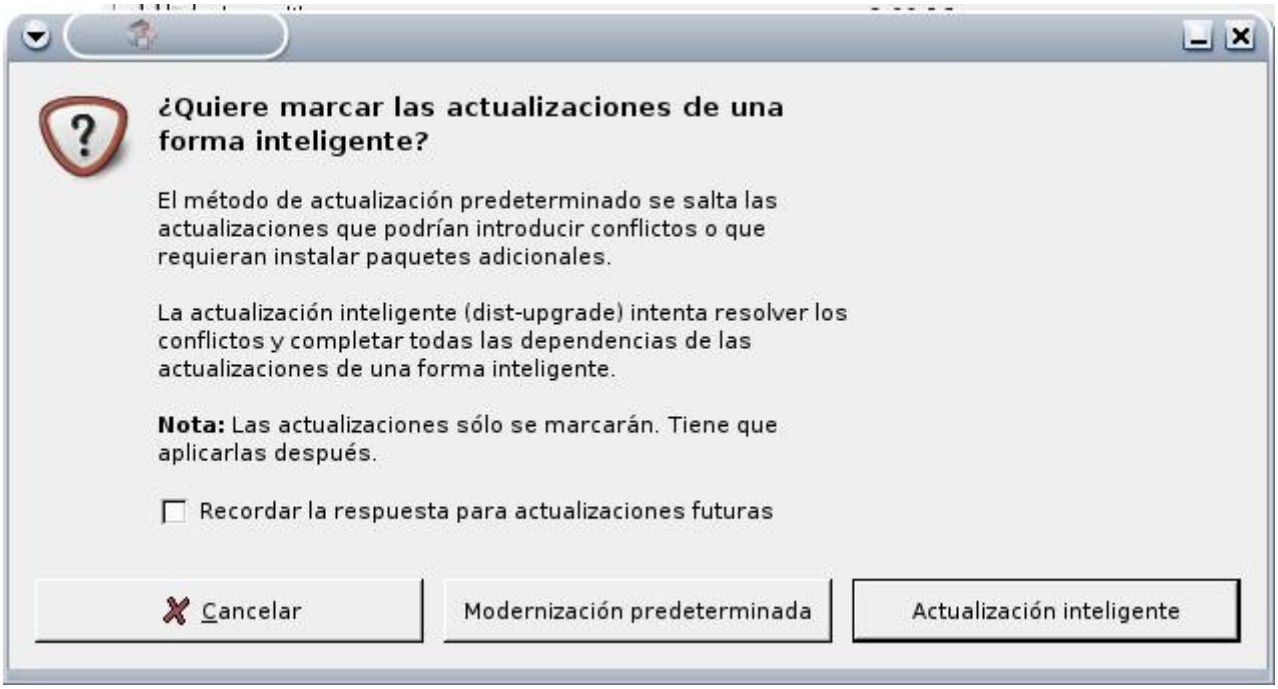
¿Y si tenemos un problema, qué?

Os acordáis de la entrega 21 donde hablábamos de problemas como el "problema de que no pueden instalarse los paquetes porque ya no se encuentran, o bien has instalado algo que no venía en los repositorios y trastes de 'dkpg' y ocurrieron dependencias, en cualquier caso siempre que se encuentren dependencias rotas."

Pues la forma de solucionarlo es primero pulsando en 'Recargar' y luego ir al menú 'Editar' y seleccionar
'Reparar paquetes rotos' (equivalente del 'apt-get -f install').

Y ahora queremos actualizar nuestro sistema.

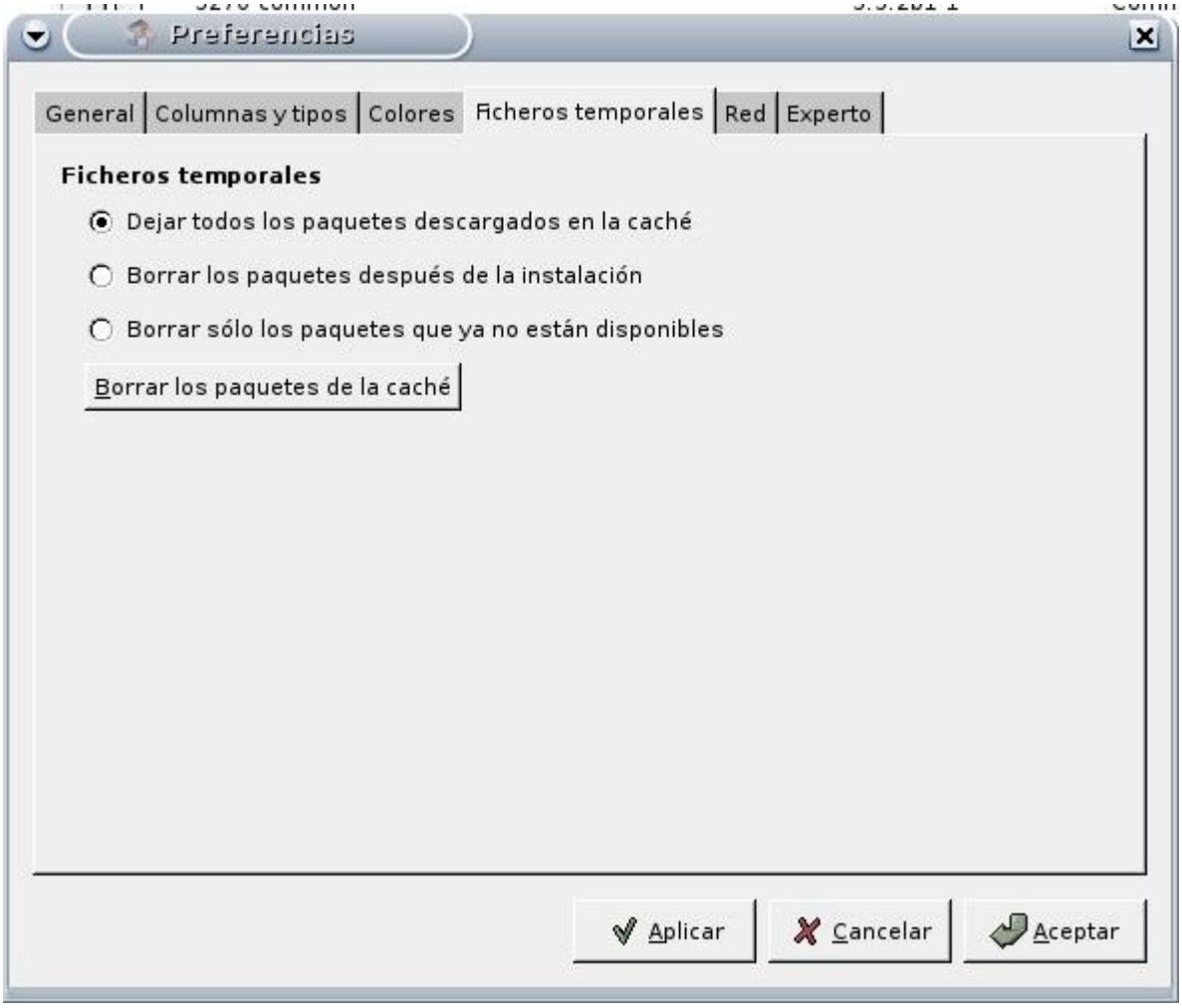
Para actualizar el sistema también es bien sencillo, pulsamos en 'Marcar todas las actualizaciones':



Y nos aparecen los 2 modos que explicamos en la entrega anterior, así que elegid la que os convenga sabiendo que 'Modernización predeterminada' equivale al 'apt-get upgrade' y 'Actualización inteligente' equivale a 'apt-get dist-upgrade'. Luego pulsáis 'Aplicar' y todo resuelto.

Y la cache, pero ¿qué pasa con la cache?.

Pues lo de la cache está más escondido pero también está, nos dirigimos al menú 'Configuración' , elegimos 'Preferencias' y vamos a la pestaña de 'Ficheros temporales':



Simplemente dejamos marcada la opción que deseamos, la segunda corresponde a 'apt-get clean' y la tercera a 'apt-get autoclean'.

Estupendo, ya sólo me queda realizar búsquedas.

Esto es fácil, y seguro que si no os lo digo también veréis ese enorme botón que pone 'Buscar':

Ya solo os queda seleccionar dónde buscar y todos nuestros problemas solucionados.

Despedida y cierre.

Con esta entrega damos por terminada esta serie dedicada a la instalación de paquetes. Si alguien

quiere hacer una entrega para 'yum' , 'yast' , 'emerge' , 'urpmi' o cualquier otro gestor estaré encantado de añadirla a la serie de entregas, porque de momento yo no voy a hacerlas. En la próxima veremos el uso básico del fichero de configuración '/etc/fstab', que sirve para montar nuestras unidades como son discos duros, pen usb, grabadoras, lectores de tarjetas, cd/dvd y demás.

¿Se anima o no se anima la cosa? Hasta la próxima entrega.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 24. Montando unidades.

```
% ^What is saccharine?
Bad substitute.
```

Preámbulo.

Según mi experiencia por los foros (sobre todo en los de www.noticias3d.com que no son precisamente específicos de Linux), hay 3 cosas que realmente se llevan la palma en cuanto a generación de dudas puesto que se repiten con mucha frecuencia. Son las siguientes: la instalación y/o compilación de programas, cómo montar dispositivos (entiéndase discos duros, pen usb, grabadoras, multilectores...), y las referentes a la configuración de las X(entorno gráfico) y conseguir la aceleración por Hardware de la tarjeta de vídeo.

La primera más o menos está ya vista o se han visto las nociones para defenderse. La segunda es lo que vamos a ver en esta entrega, cosa que aunque puede llegar a ser compleja lo intentaremos ver de una forma sencilla y práctica. Y la tercera puede ser un tema bastante avanzado que creo conveniente ver aún más adelante, sobre todo por lo extenso que puede llegar a ser.

¿Qué necesitamos saber y repasar?

Para la presente entrega es necesario saber cómo se nombran los dispositivos en Linux, cosa que ya vimos en la entrega 2 y si a estas alturas no lo sabes debes repasar la entrega 2, teniendo en cuenta que hay una errata ya que los ratones usb se referencian generalmente por '/dev/input/mice', pero concretamente dentro de /dev/usb veremos los dispositivos para esos puertos como el ratón, escáner e impresoras entre otros.

En cuanto a las unidades de almacenamiento como son discos duros, disqueteras, unidades ópticas, pen usb y demás, ya sean locales o de forma remota (a través de la red por ejemplo), saber que no se accede directamente a ellas por su dispositivo.

En los sistemas Unix no existen las unidades como en Windows (A:, B:, C: ...). Existe, como ya comentábamos en las primeras entregas, un directorio '/' a partir del cual cuelga todo. Para acceder a una unidad (como una partición) hay que montarla en un directorio dentro de nuestro árbol de directorios. Con esto conseguimos un 'todo' uniforme y accedemos a todo de la misma manera, ya sean particiones FAT (las propias de Linux) o una tarjeta de memoria o la memoria de una cámara digital, esté dentro de tu ordenador o en la China. Es importante entender esto y el por qué se montan y se desmontan unidades, que no es más que enlazar y desenlazar con el árbol de directorios, para no escuchar cosas tan absurdas como que “Linux está muy atrasado porque hay que montar unidades”.

Por convenio, consenso o costumbre, los dispositivos son montados en un directorio dentro '/mnt',y en muchas distribuciones los dispositivos removibles como disqueteras y dispositivos ópticos dentro de '/media'. Aunque realmente podemos montarlos en cualquier otro directorio que no sea '/', si bien es completamente absurdo montar unidades en sitios del sistema como '/boot' , '/bin', '/usr' y similares.

Y por último, 1 dispositivo se monta en un único directorio para él solito, nada de montar todo en el mismo sitio. Si se os ocurre montar en un directorio que no esté vacío observaréis que lo anterior desaparece, realmente lo que sucede es que queda oculto e inaccesible hasta que se desmonte el

dispositivo.

Tanto monto, monto tanto...

Para montar unidades tenemos un comando muy interesante y útil:

mount [-o Opciones] [-t tipo] [dispositivo] [directorio]

Evidentemente esta no es la única forma de utilizar mount ya que existen más opciones y parámetros, pero sí la que nos será de utilidad para nosotros que estamos aprendiendo. Lo primero que vamos a ver es el uso de mount sin parámetros:

```
matados2k@fortaleza:~$ mount
/dev/hda11 on / type ext3 (rw,errors=remount-ro)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/hda10 on /boot type ext3 (rw)
/dev/hda6 on /home type ext3 (rw)
/dev/hda12 on /usr type ext3 (rw)
/dev/hda9 on /mnt/auxi2 type vfat (rw,noexec,nosuid,nodev,umask=0)
/dev/hda8 on /mnt/auxi1 type vfat (rw,noexec,nosuid,nodev,umask=0)
/dev/hda7 on /mnt/juegos type vfat (rw,noexec,nosuid,nodev,umask=0)
/dev/hda1 on /mnt/win_xp type ntfs (ro,noexec,nosuid,nodev,umask=0)
usbfs on /proc/bus/usb type usbfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
matados2k@fortaleza:~$
```

Lo que aquí se nos muestra es lo que tenemos montado actualmenteen nuestro sistema, de la forma:

Dispositivo *on* **Directorio** *type* **Tipo Opciones**

donde el Tipo indica el sistema de ficheros que usa, y Opciones nos indica las opciones con las que fue montado, más adelante veremos qué significan.

Si estáis atentos os daréis cuenta de que existen Directorios a los que no hay asignados un dispositivo real, como por ejemplo “proc on /proc type proc (rw)”:

```
matados2k@fortaleza:~$ ls /dev/proc
ls: /dev/proc: No existe el fichero o el directorio
matados2k@fortaleza:~$
```

Esto es debido a que existen sistemas de ficheros que no están asociados a un dispositivo real, es más, en el caso '/proc' de este ejemplo ni siquiera existe físicamente, es una representación en memoria de los procesos que corren por el sistema y más información útil que nos proporciona el kernel. Pero de

momento daremos de lado esto y nos centraremos en lo que nos interesa.

El kernel de Linux es capaz de reconocer muchos sistemas de ficheros, y entre ellos los que más nos interesan son los siguientes:

vfat	Para particiones propias de Windows tipo FAT ya sea FAT16, FAT32 o FAT12 de los disquetes
ntfs	Para particiones propias de los Windows NT/200X/XP
ext2	Particiones de Linux correspondientes al antiguo estándar, pero aun muy en uso.
ext3	Particiones de Linux correspondientes al nuevo estándar
swap	Para particiones swap de Linux.
raiserfs	Particiones de tipo RaiserFs muy utilizadas en Linux
iso9660	Un sistema de ficheros local para usado para discos CD-ROM.
ufs	Un sistema de ficheros local para usado para discos CD-ROM.

Pero hay más como pueden ser minix, ext, xiafs, hpfs, msdos, umsdos, proc, nfs, smbfs, ncpfs, affs, ufs, romfs, sysv, xenix, coherent... Como veis, una gran cantidad de posibilidades imposibles de conseguir con un Windows.

Y ahora queda la gran prueba: montar un dispositivo, y aunque la mayoría de vosotros tendréis ya facilitado la forma de montarlo vamos a hacerlo manualmente. Para ello usaremos la opción '-t' que sirve para indicar el tipo de sistema a usar, yo para referenciar al cd-rom lo haré como /dev/hdXY, aunque puedes usar /dev/cdrom:

```
matados2k@fortaleza:~$ cd curso
matados2k@fortaleza:~/curso$ mk cdrom
bash: mk: command not found
matados2k@fortaleza:~/curso$ mkdir cdrom
matados2k@fortaleza:~/curso$ su
Password:
fortaleza:/home/matados2k/curso# mount -t iso9660 /dev/hdc
/home/matados2k/curso/cdrom/
mount: dispositivo de bloques /dev/hdc está protegido contra escritura; se monta
como sólo lectura
fortaleza:/home/matados2k/curso#
```

Como veis sólo root puede montar unidades, y ahora podremos ver el contenido:

```
fortaleza:/home/matados2k/curso# cd cdrom/
fortaleza:/home/matados2k/curso/cdrom# ls
autorun.bat  autorun.pif  cdrom.ico       index.html
autorun.inf  boot         change-log.txt  KNOPPIX
fortaleza:/home/matados2k/curso/cdrom#
```

Y ahora cómo saco mi CD, no me deja.

Una unidad montada nunca debe desconectarse o retirarse sin antes desmontarse, y esto muy

importante ya que Linux usa un sistema de buffers (memoria intermedia entre el dispositivo y el procesador, ya que los sistemas de almacenamiento son muchísimo más lentos que la memoria principal) en memoria que puede no estar actualizado con respecto a la unidad (aunque en este caso sea de sólo lectura), cada cierto tiempo el contenido de esos buffers es sincronizado con el dispositivo y al desmontarse lo que se hace es descargar todos esos buffers para poder retirar de forma segura el dispositivo (o apagar la máquina sin perder datos).

¿Por qué esto es así? Por razones de eficiencia, aunque esto suponga pérdida de facilidad (cosa que actualmente no es así, ya que las distribuciones más amigables están preparadaspara no tener que montar y desmontar, tal como una Mandriva por poner un ejemplo).

El comando que necesitamos es:

umount dispositivo|directorio

Y este comando es bien sencillo, para desmontarlo simplemente ejecutamos 'umount' y le indicamos o bien el dispositivo que queremos desmontar o bien el directorio.

```
fortaleza:/home/matados2k/curso/cdrom# umount /dev/hdc
fortaleza:/home/matados2k/curso/cdrom#
```

No me desmonta la unidades, ¿qué hago?

Algunas veces nos ocurre que no se desmontan las unidades, cosaque puede llegar a desesperarnos, lo primero que hay que mirar es que no esté ejecutándose nada que use lo que tenemos montado, la mayor parte de las veces lo solucionamos con esto (es evidente que si algo está en uso no lo vamos a poder quitar). Otro problema y más gordo es que un proceso ande por ahí bloqueado y no libere el dispositivo (este problema es más complicado, ya veremos en sucesivas entregas cómo ver y eliminar procesos).

Si aparece este problema contamos con un comando bastante útil que nos puede solucionar la papeleta, y su uso básico es:

eject [dispositivo|directorio]

No es necesario en este caso ser root, por defecto si no se le indica nada se referirá a '/dev/cdrom' así que vamos a ejecutarlo para nuestro ejemplo:

```
fortaleza:/home/matados2k/curso# mount -t iso9660 /dev/hdc
/home/matados2k/curso/cdrom/
mount: dispositivo de bloques /dev/hdc está protegido contra escritura; se monta
como sólo lectura
fortaleza:/home/matados2k/curso#
fortaleza:/home/matados2k/curso/cdrom# exit
exit
matados2k@fortaleza:~$ eject hdc
matados2k@fortaleza:~$
```

Como veis es tan amable que hasta nos expulsa el CD, y no es necesario poner antes '/dev/'. Por

cierto, no seáis ilusos y esperéis que os expulse un disquete ;).

Despedida y cierre.

Con el uso básico de mount y umont nos vamos a despedir de esta entrega, pero en las siguientes ya veremos cómo montar discos duros, un pen usb, un móvil (tsm 100v, pero es lo mismo que montar un pen, otro móvil y/o una cámara de fotos), una tarjeta compact flash y una tarjeta SD. Veremos las opciones más interesantes y la forma de hacernos la vida más fácil y automática con el uso de /etc/fstab y olvidarnos de tanta opción y parafernalia. Incluso por el camino veremos cómo hacer una imágenes de nuestros dispositivos y montarlas ;).

Estaremos unas cuantas entregas más tocando este tema, y posteriormente se van a tratar los siguientes:

- Monitorización y eliminación de procesos.
- Comprimir y descomprimir.
- Arranque y Parada en Linux.
- Sonido en Linux.
- Programar script.

Más o menos seguiré ese orden aunque es probable que meta temas de por medio y adelante, atrase y añada temas según me lo pidáis. Así que ya sabéis, mandadme vuestras sugerencias al foro y a mi correo.

Otra cosa importante NO RESPONDERÉ NI UNA SOLA DUDA VÍA MAIL, para eso USAD LOS FOROS que encontrareis en mi propia página, en www.sinuh.org en la sección de comunidad, también podéis encontrarme en los foros de www.noticias3d.com donde soy moderador de la sección GNU/Linux y Software Libre. De hecho ignoraré como de costumbre (salvo alguna excepción) los e-mail con dudas, por el simple hecho de que: a) ya lo avisé en la entrega 0, y b) las preguntas suelen ser muy repetitivas, y las soluciones útiles para más de una persona. El mejor sitio para poder utilizar la misma solución para más de una persona son los foros, y si lo preguntan 50 veces con responderla 1 sobra. Tampoco ESPERÉIS RESPUESTA EN LOS COMENTARIOS, ya que tampoco es el sitio indicado. Espero que sepáis comprender que estar resolviendo las mismas dudas una y otra vez vía e-mail no es nada agradable y poco útil salvo para la persona que pregunta.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 25. Montando unidades (II).

```
% ^What is saccharine?
Bad substitute.
```

Continuando.

Antes de nada, vamos a ver cómo desaparecen los archivos dentro de un directorio que usamos para montar una unidad y cómo especificar que intente adivinar qué tipo de sistema de ficheros usa :).

```
matados2k@fortaleza:~$ cd curso
matados2k@fortaleza:~/curso$ ls -l > ./cdrom/desaparezco.txt
matados2k@fortaleza:~/curso$ su
Password:
fortaleza:/home/matados2k/curso# cd cdrom/
fortaleza:/home/matados2k/curso/cdrom# ls
desaparezco.txt
fortaleza:/home/matados2k/curso/cdrom# mount -t auto /dev/hdc
/home/matados2k/curso/cdrom/
mount: dispositivo de bloques /dev/hdc está protegido contra escritura; se monta
como sólo lectura
fortaleza:/home/matados2k/curso/cdrom#
fortaleza:/home/matados2k/curso/cdrom# ls
desaparezco.txt
fortaleza:/home/matados2k/curso/cdrom# cd ..
fortaleza:/home/matados2k/curso# cd cdrom
fortaleza:/home/matados2k/curso/cdrom# ls
autorun.bat  autorun.inf  cdrom.ico  info  isolinux  META
fortaleza:/home/matados2k/curso/cdrom#
```

Como vemos en el ejemplo, creamos un archivo de texto dentro del directorio donde vamos a montar el cdrom, pero en este caso usamos 'auto' con lo que le indicamos que autodetecte el formato, ¡bien! esto se empieza a facilitar, aunque no siempre funciona. Como ya estábamos dentro del directorio antes de montarlo, aún es visible el archivo, pero es salir y entrar y ¡oops! ya no está. Que no cunda el cúnico, esto tiene fácil solución ;)

```
fortaleza:/home/matados2k/curso/cdrom# umount /dev/hdc
umount: /home/matados2k/curso/cdrom: dispositivo ocupado
umount: /home/matados2k/curso/cdrom: dispositivo ocupado
fortaleza:/home/matados2k/curso/cdrom# cd ..
fortaleza:/home/matados2k/curso# umount /dev/hdc
fortaleza:/home/matados2k/curso# cd cdrom
fortaleza:/home/matados2k/curso/cdrom# ls
desaparezco.txt
fortaleza:/home/matados2k/curso/cdrom#
```

Como veis, es imposible desmontar si estamos dentro, puesto que evidentemente está en uso. Simplemente desmontando ya volveremosa tener los ficheros originales.

Opcionalmente las Opciones.

Muchas veces nos puede resultar necesario usar opciones para el tipo de sistema de archivos que vamos a montar, y opciones válidas para todos.

En nuestro ejemplo enparticular, lo ideal sería no ver: 'mount: dispositivo de bloques /dev/hdc está protegido contra escritura; se monta como sólo lectura' diciéndole que lo monte directamente en modo de sólo lectura:

```
fortaleza:/home/matados2k/curso/cdrom# mount -o ro -t auto /dev/hdc
/home/matados2k/curso/cdrom/
fortaleza:/home/matados2k/curso/cdrom# umount /dev/hdc
fortaleza:/home/matados2k/curso/cdrom#
```

La opción 'ro' significa 'read only', o lo que es lo mismo, montar en modo sólo lectura. Ahora veremos qué tipos de opciones interesantes tenemos:

NOTA: Algunas opciones no tienen sentido fuera del fichero de configuración '/etc/fstab', que veremos más adelante en esta serie de entregas.

Válidas para todos los sistemas:	
auto	Puede montarse con la opción -a (no confundir con indicar en el sistema de archivos 'auto'). Más adelante veremos para qué sirve la opción '-a'.
defaults	Emplear las opciones predeterminadas.
exec	Permitir la ejecución de binarios.
noatime	No actualizar los tiempos de acceso a nodo -í en este sistema de ficheros. A cambio de no actualizar los tiempos de acceso de un fichero podemos ganar velocidad.
noauto	Sólo puede montarse explícitamente (esto es, la opción -a no hará que el sistema de ficheros se monte).
noexec	No permitir la ejecución de ningún binario en el sistema de ficheros montado.

Válidas para todos los sistemas:	
nosuid	No permitir el efecto de los bits SUID ni SGID.
nouser	Prohibir a un usuario ordinario (esto es, distinto de root) montar el sistema de ficheros. Esto es lo predeterminado.
remount	Intentar re-montar un sistema de ficheros ya montado. Esto se emplea comúnmente para cambiar las opciones de montaje en un sistema de ficheros, especialmente para que se pueda escribir en un sistema de ficheros que estaba de lectura exclusiva.
ro	Montar el sistema de ficheros de lectura exclusiva.
rw	Montar el sistema de ficheros de lectura y escritura.
suid	Permitir el efecto de los bits SUID y SGID.
user	Permitir a un usuario ordinario montar el sistema de ficheros.

Estas son las que normalmente podéis usar (y me quedo unas cuantas en el tintero), pero veremos algunas más que os pueden interesar:

Válida para Ext2	
check / check=normal / check=strict	Establece el nivel de comprobaciones. Esto hace que la integridad de los datos se comprueben cuando se monta la partición, más información 'man mount'
Válida para tipo FAT	
uid=valor	Establece el propietario de todos los ficheros
gid=valor	Establece el grupo de todos los ficheros.
umask=valor	Establece la umask (la máscara de bits de los permisos que no están presentes).

Con estas opciones tendremos más que suficiente para saber movernos montando y desmontado particiones. Como podéis intuir, esta cantidad de opciones nos da una flexibilidad que un sistema Windows por ejemplo no podría imaginar

Montando discos duros.

Ahora veremos cómo montar las particiones típicas de Windows que muchos de nosotros tenemos, para empezar las más fáciles: las fat16 (en extinción) y fat32. Muchos de vosotros ya las tendréis montadas, así que para las pruebas desmontadlas antes:

```
fortaleza:/home/matados2k/curso# umount /mnt/win_xp
fortaleza:/home/matados2k/curso# umount /dev/hda7
fortaleza:/home/matados2k/curso#
```

La primera que desmonta es una partición ntfs mediante el directorio donde estaba montada, y la segunda es una fat32 mediante su dispositivo.

Vamos a montar seguidamente la partición fat32 en modo lectura y escritura, y hacer que todos la puedan utilizar poniendo la máscara de permisos a 0.

```
fortaleza:/home/matados2k/curso# mount -o rw,umask=0 -t vfat /dev/hda7
/mnt/juegos
fortaleza:/home/matados2k/curso#
```

Fácil, ¿verdad? Para poner más de una opción ya sabéis, poned una coma. Ahora lo complicamos: la vamos a remontar con las mismas características, que no permita la ejecución de binarios, que autodetecte qué tipo de partición es y en sólo lectura:

```
fortaleza:/home/matados2k/curso# mount -o remount,ro,umask=0,noexec -t auto
/dev/hda7 /mnt/juegos
fortaleza:/home/matados2k/curso#
```

Podéis complicarlo tanto como queráis. Ahora veremos cómo montar una partición ntfs típica de los Windows NT/200X/XP. Éstas son más problemáticas, ya que el núcleo de Linux muchas veces no está compilado con soporte para ellas (depende de la distribución que uséis), y oficialmente no hay soporte para la escritura (por mucho que os digan que sí, sólo hay que leer la documentación del kernel para verlo), así que montadlas en modo sólo lectura para evitaros problemas. Para aquellos que vuestro kernel no venga con soporte ntfs lo siento, pero no podemos ver a estas alturas cómo conseguirlo. Actualmente se puede acceder a las particiones NTFS en modo escritura de forma bastante segura con el driver ntfs-3g, pero tampoco vamos a ver cómo usarlo.

```
fortaleza:/home/matados2k/curso# mount -o ro,umask=0 -t ntfs /dev/hda1
/mnt/win_xp
fortaleza:/home/matados2k/curso# mount -o remount,ro,umask=0 -t auto /dev/hda1
/mnt/win_xp
fortaleza:/home/matados2k/curso#
```

Ahí tenéis el ejemplo de cómo las monta sin mayor problema. Y como nota, las particiones tipo Ext2 y Ext3 montadlas sólo con la opción 'default', que os funcionarán sin problemas.

Montando unidades externas por usb.

Las unidades externas por usb como son pen drives, lectores de tarjetas de memoria, cámaras digitales (no todas, pero sí la mayoría) y demás tienen un trato un tanto especial, ya que Linux las monta como dispositivos scsi ('/dev/sdXY', donde X e Y tienen el mismo significado que para '/dev/hdXY').

Según vas pinchando se les va asignando un dispositivo '/dev/sdX' que se corresponde al primero que esté libre (yo no tengo nada scsi, ni discos duros sata reconocidos como scsi, así que yo empiezo por

'/dev/sda'), en mi caso particular tengo un lector de tarjetas de memoria de estos que van en el hueco de la disquetera, concretamente con 4 ranuras para 6 tipos de tarjetas, entonces como es el primer dispositivo usb que encuentra el ordenador al arrancar me coge desde '/dev/sda' a '/dev/sdd', uno por cada ranura. Lo siguiente que conecte irá a partir de '/dev/sde'.

Como todos los dispositivos usb de almacenamiento del tipo que comento son internamente de tipo FAT, todos se montan de igual manera, la misma que los discos duros. ¡Pero! (ya empezamos, si es que no puede ser tan bonito) al igual que los discos duros la mayoría de éstos tienen una partición, generalmente sólo una, con lo que por ejemplo si es '/dev/sda' al montarlo le indicamos '/dev/sda1' para indicarle la primera partición primaria de '/dev/sda', esto será lo mas normal y raramente nos encontraremos (al igual que con los disquetes) con unidades que no tengan particiones con lo cual nos olvidemos del número y tan contentos :).

Así que primero pasaremos a montar un pen usb, como yo ya sé que me lo asignará sobre /dev/sde lo tengo más fácil, pero claro... y los demás qué, ¿prueba y error? Pues es una solución, pero mejor usemos un comando muy útil llamado:

dmesg

Este es un comando que muestra el 'ring buffer' (no se bien cómo traducirlo, así que lo dejo tal cual y que cada uno haga su traducción). Y de momento lo vamos a usar para lo que nos interesa, que es ver dónde asigna Linux nuestro pen. Para esto lo pinchamos e inmediatamente ejecutamos este comando:

```
matados2k@fortaleza:~$ dmesg
...
usb-storage: device found at 5
usb-storage: waiting for device to settle before scanning
  Vendor:           Model: USB DISK Pro      Rev: PMAP
  Type:   Direct-Access                      ANSI SCSI revision: 00
SCSI device sde: 489472 512-byte hdwr sectors (251 MB)
sde: assuming Write Enabled
sde: assuming drive cache: write through
SCSI device sde: 489472 512-byte hdwr sectors (251 MB)
sde: assuming Write Enabled
sde: assuming drive cache: write through
 /dev/scsi/host2/bus0/target0/lun0: p1
Attached scsi removable disk sde at scsi2, channel 0, id 0, lun 0
usb-storage: device scan complete
matados2k@fortaleza:~$
```

Si somos observadores nos daremos cuenta de que estos son todos los mensajes que Linux nos daba al arrancar, y en lo último de todo aparece lo que acabamos de pinchar, nuestro pen usb. Nos fijamos en esta línea:
'`SCSI device sde: 489472 512-byte hdwr sectors (251 MB)`'
que es precisamente la que nos ha chivado dónde esta :). Así que sin más:

```
matados2k@fortaleza:~$ cd curso/
matados2k@fortaleza:~/curso$ mkdir usb
matados2k@fortaleza:~/curso$ su
Password:
fortaleza:/home/matados2k/curso# fortaleza:/home/matados2k/curso# mount -t vfat
-o rw,umask=0 /dev/sde ./usb
mount: wrong fs type, bad option, bad superblock on /dev/sde,
       missing codepage or other error
       In some cases useful info is found in syslog - try
       dmesg | tail  or so

fortaleza:/home/matados2k/curso# mount -t vfat -o rw,umask=0 /dev/sde1 ./usb
fortaleza:/home/matados2k/curso#cd usb
fortaleza:/home/matados2k/curso/usb# ls
cd  seguridad
fortaleza:/home/matados2k/curso/usb#
```

Como veis mi pen usb tiene una partición y accedo por /dev/sde1. Pues es así con todos los dispositivos de almacenamiento por usb, como mi lector de tarjetas con una SD:

```
fortaleza:/home/matados2k/curso/usb#dmesg
...
sdc: Spinning up disk....ready
SCSI device sdc: 499712 512-byte hdwr sectors (256 MB)
...
fortaleza:/home/matados2k/curso/usb#cd ..
fortaleza:/home/matados2k/curso# umount ./usb
fortaleza:/home/matados2k/curso# mount -t vfat -o rw,umask=0 /dev/sdc1 ./usb
fortaleza:/home/matados2k/curso# cd usb
fortaleza:/home/matados2k/curso/usb#
```

O como un móvil, una cámara digital (la mayoría)... con todos es exactamente igual, por lo que más ejemplos sobran ;).

Despedida y cierre

En la próxima entrega veremos cómo automatizar todo esto y facilitarnos la vida con el fichero '/etc/fstab'.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 26. Montando unidades (III).

% [Where is my brain?
Missing].

El fichero /etc/fstab.

Este es un fichero de configuración que nos interesa muchísimo, ya que guarda información estática sobre los sistemas de ficheros. Cuando nuestro sistema arranque leerá este fichero de configuración y actuará en consecuencia, montando todo aquello que le indiquemos o guardando datos para simplificar el montado posterior.

Sin más, abrimos el que tengamos cada uno en nuestro sistema, el mío es tal que así:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda11      /               ext3    defaults,errors=remount-ro 0       1
/dev/hda10      /boot           ext3    defaults        0       2
/dev/hda6       /home           ext3    defaults        0       2
/dev/hda12      /usr            ext3    defaults        0       2
/dev/hda5       none            swap    sw              0       0
/dev/hdc        /media/cdrom0   iso9660 ro,user,noauto  0       0
/dev/hdd        /media/cdrom1   iso9660 ro,user,noauto  0       0
/dev/hdb1       /mnt/m2kaux     vfat    rw,user,umask=0 0       0
/dev/hda1       /mnt/winxp      ntfs    ro,user,umask=0 0       0
/dev/hda7       /mnt/juegos     vfat    rw,user,umask=0 0       0
/dev/hda8       /mnt/auxi1      vfat    rw,user,umask=0 0       0
/dev/hda9       /mnt/auxi2      vfat    rw,user,umask=0 0       0
/dev/sda1       /mnt/cfmd       vfat    rw,user,umask=0 0       0
/dev/sdc1       /mnt/sdmmc      vfat    rw,user,umask=0 0       0
/dev/sde1       /mnt/usb        vfat    rw,user,umask=0 0       0
```

Todo lo que veamos detrás de un carácter '#' son líneas comentadas y no tendrán efecto alguno, con lo que tenemos que en la 3ª línea de mi fichero (no tiene por qué ser así en el del resto) está comentado el formato que veremos a continuación:

<file system><mount point><type><options><dump><pass>

Significado
"File system", o sea sistema de fichero, se corresponde con el dispositivo que usábamos en el

Significado
comando mount (ver entrega 24).
“Mount point”, punto de montaje, es el directorio donde queremos montar el dispositivo.
“Type”, tipo, se corresponde a la opción -t del comando mount.
“options”, opciones, se corresponde con lo que poníamos para la opción -o de mount.
“dump”, sólo es necesario si usamos el comando 'dump' para determinar qué sistemas necesitan ser volcados, siendo dump una herramienta de backup. Lo normal es dejarlo a 0 si no lo tenemos instalado o no vamos a hacer uso de él, que suele ser lo normal.
“pass”, lo usa el programa fsck (este ya si que es más importante). Este programa realiza un chequeo al arrancar de los sistemas de ficheros, y el número indica en qué orden deben comprobarse, teniendo en cuenta que los que tengan el mismo número se comprueban de forma secuencial a como han sido definidos. Hay que tener en cuenta que si los sistemas se encuentran en distintos dispositivos se hace una comprobación concurrente, aprovechando así el paralelismo del sistema. Si este número es un 0 (lo normal) se entiende que no hay que comprobarlo. El manual (comando 'man') recomienda en caso de que se comprueben, que a la partición raíz ('/') se le asigne un 1 y al resto un 2.

Visto esto y sabiendo manejar el comando mount, ya es cosa trivial saber cómo usar este fichero, para ayudarnos podemos usar la tabla de la entrega 25 y ya nada se nos debe resistir.

Con esto ya sabemos configurar este fichero, y ya sabemos que todo lo que definamos en él se monta automáticamente al iniciar el sistema, salvo que le digamos lo contrario como en las líneas 10 y 11 de mi fichero, donde si os fijáis en las opciones aparece 'noauto' (una de las que aparece en las tablas de la entrega 25, y que no tiene sentido fuera de este fichero). ¿Entonces, de qué nos vale? Pues muy sencillo, puesto que todo se simplifica a la hora de montar unidades. Ejemplos:

```
matados2k@fortaleza:~$ mount /dev/cdrom1
matados2k@fortaleza:~$ ls /media/cdrom1/
autorun.inf   devel                 dyne.png      isolinux       logo.png
ChangeLog     dyne                  extras        LICENSE.TXT    README.TXT
CLICK_ME.HTM  dynebolic-manual.pdf  floppy        linuxboot.cfg
default.xbe   dynebol.ico           gnulinux.png  loadlin
matados2k@fortaleza:~$
```

No he tenido nada más que indicarle el dispositivo, y no sólo eso, sino que además he podido hacerlo como usuario normal ya que definí la opción 'user' para este dispositivo en mi fichero, permitiendo que cualquier usuario pueda montarla unidad (todas las opciones fueron ya explicadas en las entrega 25).

```
matados2k@fortaleza:~$ umount /media/cdrom1
matados2k@fortaleza:~$ mount /mnt/usb
matados2k@fortaleza:~$ ls /mnt/usb/
instalador      Modserver2.sxw  test1.jpg  test3.jpg  test5.jpg  test7.jpg
Modserver2.pdf  modserver.sxi   test2.jpg  test4.jpg  test6.jpg  test8.jpg
matados2k@fortaleza:~$ umount /dev/sde1
matados2k@fortaleza:~$
```

Podemos observar entonces que podemos montar y desmontar todo lo definido en /etc/fstab con sólo indicar su dispositivo o el directorio a montar. ¿No es estupendo? Y que desmontar sí sabíamos que se podía de ambas formas. Y además hemos visto que se puede hacer que otros usuarios monten unidades además de root, ya nos vamos acercando a una configuración ideal ;).

Y si os fijáis, las 3 últimas líneas de mi fichero son las que montan las cosas que conecto al usb y al lector de tarjetas. ¿No sería lógico entonces añadirles la opción 'noauto' para que no intente añadirlas al principio? Pues sí, pero... podemos ver otra ventaja:

```
matados2k@fortaleza:~$ su
Password:
fortaleza:/home/matados2k# mount -a
mount: el dispositivo especial /dev/sda1 no existe
mount: el dispositivo especial /dev/sdc1 no existe
mount: el dispositivo especial /dev/sde1 no existe
fortaleza:/home/matados2k# mount -a
mount: el dispositivo especial /dev/sda1 no existe
mount: el dispositivo especial /dev/sdc1 no existe
fortaleza:/home/matados2k#
```

Vemos aquí una nueva opción de mount que es '-a', que intenta montar automáticamente todo lo definido en /etc/fstab si no lo estaba ya o no se le indica lo contrario con 'noauto', y como veis los 3 errores primeros son debidos a que no hay nada conectado (ya que los ha intentado montar por no tener puesto 'noauto'). Pero voy y pincho un pen usb, y mira por donde, no he tenido que preocuparme de más.

¡OJO!: La opción '-a' también existe para 'umount' y os podéis imaginar qué pasa ¿no?:

```
fortaleza:/home/matados2k# umount -a
umount: /dev: dispositivo ocupado
umount: /mnt/auxi1: dispositivo ocupado
umount: /usr: dispositivo ocupado
umount: /home: dispositivo ocupado
umount: /boot: dispositivo ocupado
umount: /: dispositivo ocupado
fortaleza:/home/matados2k# mount -a
mount: el dispositivo especial /dev/sda1 no existe
mount: el dispositivo especial /dev/sdc1 no existe
mount: el dispositivo especial /dev/sde1 no existe
fortaleza:/home/matados2k#
```

Pues sí, lo intenta desmontar todo, y gracias a Dios si algo está en uso no se desmonta y podemos volver a poner en funcionamiento todo con 'mount -a'.

Sigo con problemas, ni 'eject' hace que se me desmonte algo.

Pues si el dispositivo sabéis de sobra que no está ocupado, como ya explicamos en las entregas anteriores, sí que es un problema y es que algo pasa. Pero si para algo es bueno Linux es para encontrar la solución para todo. Por desgracia, los usuarios de KDE solíamos tener este problema más frecuentemente de lo que desearíamos, veamos cómo combatirlo:

```
matados2k@fortaleza:~$ umount /mnt/usb/
umount: /mnt/usb: dispositivo ocupado
umount: /mnt/usb: dispositivo ocupado
matados2k@fortaleza:~$
```

Esto es un problema, porque yo sé que no tengo ninguna aplicación abierta que lo use y que está inactivo, ya que mi pen usb tiene una lucecita queahora está apagada. Algún proceso debe haberse quedado atontado mientras lo usaba, vamos a buscarlo:

```
matados2k@fortaleza:~$ su
Password:
fortaleza:/home/matados2k# lsof +D /mnt/usb
COMMAND  PID      USER   FD   TYPE DEVICE  SIZE NODE NAME
famd    7818 matados2k   89r   DIR   8,65 16384    1 /mnt/usb
famd    7818 matados2k   92r   DIR   8,65  4096  609 /mnt/usb/instalador
fortaleza:/home/matados2k#
```

Aquí tenemos nuestro culpable, sí señor, podéis señalarle con el dedo. Este comando no vamos a explicarlo ahora ni los siguientes que veamos, ya que son para una próxima serie de entregas, así que tenéis que hacer un acto de fe, ya sabéis que para encontrar el culpable es 'lsof +D aquí_dispositivo_o_directorio".

Sólo puede quedar uno, forastero.

Ahora tendríamos que eliminar ese proceso, lo normal es que el que ahí sale sea el culpable, o sea 'famd', pero yo ya sé por experiencia que no lo es. Si lo eliminamos solucionaríamos el problema, y será así normalmente, pero eliminar un servicio como 'famd' ('man famd') puede traer otros problemas.

Así que como norma general el que nos muestre ese comando es el que tenemos que eliminar, pero hay casos como éste en que no es el verdadero culpable. ¿Eso quién te lo da? Pues simplemente la experiencia. Como sé que es problema de KDE y en concreto de un proceso llamado ' konqueror' que se ha quedado atontado (aunque ya lo haya cerrado), lo paso a eliminar buscándolo primero con otro comando del que también tendréis que haceracto de fe:

```
fortaleza:/home/matados2k# ps aux
....
1000      8869 15.8  4.2  35996 21896 ?        S    19:37   0:00 konqueror [kdei
1000      8870  0.2  2.0  24688 10404 ?        S    19:37   0:00 kio_file [kdein
1000      8871  0.0  1.9  24320 10208 ?        S    19:37   0:00 kio_devices [kd
root      8872  0.0  0.1   2780   912 pts/3    R+   19:37   0:00 ps -aux
fortaleza:/home/matados2k#
```

De hecho yo sé que puede ser uno de estos 3 (ya me ha pasado muchas veces y la experiencia así me lo dice). Así que lo eliminamos (otro acto de fe):

```
fortaleza:/home/matados2k# kill 8869
fortaleza:/home/matados2k# umount /mnt/usb
fortaleza:/home/matados2k#
```

El número pasado a kill es el PID del proceso y nos lo indica tanto 'lsof +D' como 'ps aux' si os fijáis bien, tanto si hubiéramos matado 'famd' como este último solucionaríamos el problema y podríamos desmontar con seguridad nuestro pen usb.

Una es una forma más automática, y otra la dicta la experiencia de verse muchas veces con el problema, así que esto queda simplemente como ejemplo. Y una vez terminada esta serie de montado de unidades veremos con detalle estos comandos (mejor esto y poder intentar solucionar el problema que esperar a más adelante ¿no?).

A Dios pongo por testigo que si todo falla, no me rendiré. A la desesperada.

Murphy es muy puñetero y aún así puede que la cosa se resista a desmontarse como Dios manda, si algo puede fallar ya se sabe que fallará y antes que volver a reiniciar nos queda un último intento:

umount -l [dispositivo|directorio]

Esta opción '-l' es que lo desmonte por narices y eso puede tener su riesgo, de hecho (por lo menos a mí) no viene documentado en 'man umount', pero siempre es un último recurso que nos puede solucionar la papeleta:

```
matados2k@fortaleza:~$ umount /mnt/usb/
umount: /mnt/usb: dispositivo ocupado
umount: /mnt/usb: dispositivo ocupado
matados2k@fortaleza:~$ umount -l /mnt/usb/
matados2k@fortaleza:~$
```

Con esto ya hemos visto todas las soluciones que yo conozco, siempre mejores que simplemente desconectar, con lo que damos por terminada esta espesa entrega.

Despedida.

Ya sabemos prácticamente todo lo necesario para movernos con soltura montando y desmontando unidades, y hemos visto cómo simplificarlo, pero aún podemos quedar las cosas de forma más automática, y para eso la siguiente entrega.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 27. Montando unidades (y IV).

% sleep with me
bad character

Automatizando.

Ahora llega el momento de saber cómo montar y desmontar de forma más cómoda y automática, esto para los puristas suele ser un problema de seguridad, pero en un ordenador para uso propio esto se traduce en facilidad.

Para conseguir automatizar nuestro montaje de unidades tenemos dos opciones importantes que son supermount y automount. El primerolo vamos a descartar por la sencilla razón de que hay que parchear el kernel y compilarlo (según estuve viendo, si me equivoco hacédmelo saber) en el caso de que no nos venga por defecto en nuestra distribución, con lo que queda un poco lejos del nivel del curso actualmente, mientras que el segundo lo podemos usar simplemente instalando un paquete.

Ya que con ambos vamos a conseguir lo que necesitamos, nos preparamos para usar automount.

Y antes de instalar, verificamos...

Ahora vamos a comprobar que nuestro kernel tiene soporte para automount de la siguiente forma:

```
fortaleza:/home/matados2k# cat /proc/filesystems
nodev   sysfs
...
nodev   mqueue
        ext3
        vfat
        ntfs
nodev   usbfs
nodev   autofs
nodev   binfmt_misc
fortaleza:/home/matados2k#
```

Con esto obtenemos un listado más o menos amplio de todos los sistemas de ficheros soportados en nuestro kernel, y buscaremos '`nodev autofs`' , si lo tenemos pues estupendo, si no intentaremos lo siguiente:

```
fortaleza:/home/matados2k# modprobe autofs
FATAL: Error inserting autofs (/lib/modules/2.6.10-1-
k7/kernel/fs/autofs/autofs.ko): Device or resource busy
fortaleza:/home/matados2k#
```

Modprobe es un comando para cargar módulos en el kernel, pero este es un tema que no tocaremos ahora. En mi caso falla porque yo sí tengo ya autofs, si aún así seguís sin tenerlo lo mismo al instalar

el paquete 'autofs' en el siguiente punto lo consigáis, en caso contrario deberíais compilar el kernel, con lo que tendréis que dejar esta entrega para un futuro.

Instalando.

En el caso de distribuciones basadas en Debian o con un port de apt-get simplemente necesitamos lo siguiente:

```
matados2k@fortaleza:~$ su
Password:
fortaleza:/home/matados2k# apt-get install autofs
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
0 actualizados, 1 se instalarán, 0 reinstalados, 0 para eliminar y 0 no
actualizados.
Se necesita descargar 0B/106kB de archivos.
Se utilizarán 0B de espacio de disco adicional después de desempaquetar.
¿Desea continuar? [S/n] s
....
fortaleza:/home/matados2k#
```

En caso de usar rpm y no tengamos un port de apt-get, buscamos el paquete necesario en los cd's de nuestra distribución, llamado 'autofs' con alguna coletilla de versión, y en caso de no encontrarlo siempre podemos recurrir a la lista de direcciones que dimos en la entrega 19.

Ya estoy preparado, ¿y ahora qué?

Ya tenemos todo listo, y ahora veremos qué hay que configurar. Dependiendo de nuestra distribución de GNU/Linux se nos pueden haber instalado más o menos ficheros y directorios como pudiera ser '/misc', que en este caso si no vais a utilizarlo podéis eliminarlo (en mi caso no se ha creado).

Se nos hayan instalado unos u otros hay un ficheroque es el que vamos a usar y el que configura lo que necesitamos: '/etc/auto.master'. En mi caso, su contenido por defecto es el siguiente:

```
#
# $Id: auto.master,v 1.4 2005/01/04 14:36:54 raven Exp $
#
# Sample auto.master file
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# For details of the format look at autofs(5).
#/misc/etc/auto.misc --timeout=60
#/smb /etc/auto.smb
#/misc/etc/auto.misc
#/net /etc/auto.net
```

No hay definido nada, es más, está todo comentado (todo lo que aparezca detrás de una '#' es un comentario, por lo cual no tendrá efecto alguno). En este fichero definiremos todos los directorios que queramos que tengan sistemas de ficheros automontados, de la siguiente forma:

Directorio Fichero_De_Configuración [--timeout X]

En 'Directorio' añadimos el directorio que vamos a gestionar con automount, y mucho cuidado, porque todos los subdirectorios definidos dentro ya no serán visibles, más que nada por si se os ocurre decir /mnt y queréis tener sistemas que no montara automount.

En 'Fichero_De_Configuración'le indicamos en qué fichero de configuración definiremos las opciones con las que debe ser montado.

Y por último y opcionalmente '--timeout X', donde X será el tiempo en segundos que pasará un sistema sin usarse antes de ser desmontado automáticamente.Por defecto, si no se indica nada creo que está definido para 5 minutos.

Pero, ¿qué es conveniente y no conveniente que se automonte? Pues yo os aconsejaría que sólo los dispositivos removibles como disquetes, cd-roms, dvd, pen usb ... pero podéis poner lo que queráis. Sin embargo, si hay una partición que se mira de higos a brevas puede ser una buena idea que se automonte cuando se necesite (un dispositivo se monta en el momento que se intenta acceder a él) en vez de estar montado siempre. Así liberaríamos recursos.

En mi caso voy a preparar mi grabadora de cd's ,mi dvd y mi pen usb, y usaré mi directorio /media para ello, así que me defino lo siguiente:

```
/media/cdrom0 /etc/auto.cdrom0 --timeout 15

/media/cdrom1 /etc/auto.cdrom1 --timeout 15

/media/usb    /etc/auto.usb --timeout 5
```

o bien

```
/media                  /etc/auto.media  --timeout 5
```

En la primera opción defino los directorios por separado con un fichero de configuración para cada uno, y en el segundo defino un directorio base y en el fichero de configuración 'auto.media' definiré los directorios. Para quienes quieran montar bajo '/mnt' sin cargarse el resto de los sistemas no gestionados por autofs mejor que usen la primera opción, yo optaré por la segunda.

Ahora debemos crear los ficheros de configuración definidos, como he optado por la segunda sólo deberé crear uno '/etc/auto.media', en él definiremos cada directorio dentro de '/media', y cómo será montada y su dispositivo (para la primera opción el nombre del directorio sin la ruta, cómo será montada y su dispositivo).

El formato de la línea es el siguiente:

Directorio [-opciones] :dispositivo

En mi caso queda de la siguiente forma:

```
cdrom0   -fstype=auto,ro,user,noauto :/dev/hdc
cdrom1   -fstype=auto,ro,user,noauto :/dev/hdd
usb      -fstype=vfat,rw,user,umask=0 :/dev/sde1
```

Con lo que observamos que no hace falta la ruta, para indicar el tipo de fichero usamos '-fstype=' y el resto de opciones separadas por comas son las mismas que el comando mount, estupendo.

Y para finalizar...

Ya sólo nos queda comentar las líneas de nuestro '/etc/fstab' referentes a lo definido para autofs, y reiniciar autofs de la siguiente forma:

```
fortaleza:/home/matados2k# /etc/init.d/autofs restart
Stopping automounter: done.
Starting automounter: done.
fortaleza:/home/matados2k#
```

Esto lo veremos próximamente por lo que no explicaré ahora qué es lo que realmente he hecho. Y a probarlo :D

```
fortaleza:/home/matados2k# exit
exit
matados2k@fortaleza:~$ cd /media/usb
matados2k@fortaleza:/media/usb$ ls
Archivo de texto
Archivo de texto~
...
matados2k@fortaleza:/media/usb$ cd /media/cdrom1
matados2k@fortaleza:/media/cdrom1$ ls
autorun.inf   devel                 dyne.png      isolinux       logo.png
ChangeLog     dyne                  extras        LICENSE.TXT    README.TXT
CLICK_ME.HTM  dynebolic-manual.pdf  floppy        linuxboot.cfg
default.xbe   dynebol.ico           gnulinux.png  loadlin
matados2k@fortaleza:/media/cdrom1$
```

Perfecto :). Pero tened en cuenta que si no definís tiempo de desmontado, tendréis que esperar el tiempo por defecto o desmontar a mano desde root. Si teníais en vuestro escritorio accesos directos a vuestros dispositivos mediante el '/etc/fstab' (como en KDE) debéis sustituirlos por accesos a los directorios y si ponéis un tiempo bajo como 5 seg de desmontado tendréis que esperar muy poco para sacar el dispositivo sin riesgo. Chúpate esa, 'Desconectar con seguridad' (Windows).

Pero aún tengo una píldora más para vosotros, montemos imágenes *.iso

Ya hemos visto todo lo necesario de esta serie para el montado y desmontado de unidades, pero por qué no aprovechar y aprender a montar esas imágenes que nos bajamos o creamos nosotros mismos en formato 'iso' sin herramientas auxiliares y con el comando 'mount' (anda, esto creo que Windows no lo hace por sí mismo).

Para ello necesitamos tener en el kernel una cosa llamada "soporte loopback" activada (normalmente estará), así que si no os funciona ya sabéis que no podréis por este motivo, hasta que aprendamos a compilarnos nuestro propio kernel.

Empezamos creándonos una imagen '.iso' de un cd nuestro, sin anticopia, por supuesto (más que nada porque se os puede quedar la unidad leyendo indefinidamente, y al cabo de un tiempo sacar el cd/dvd tan caliente que podréis moldearlo y crear arte contemporáneo).

```
matados2k@fortaleza:~$ cat /dev/hdd > mycd.iso
matados2k@fortaleza:~$
```

¿Cómo? ¿Con un simple cat a un dispositivo óptico redirecconado a un fichero, obtenemos una imagen iso? Esto es la leche.

Pues bien, ahora ya sólo nos queda usar el comando 'mount' de la siguiente forma:

mount -t iso9660 -o loop fichero.iso /ruta/montaje

Y veamos cómo me funcionó a mí:

```
matados2k@fortaleza:~$ cd curso
matados2k@fortaleza:~/curso$ mkdir iso
matados2k@fortaleza:~/curso$ su
Password:
fortaleza:/home/matados2k/curso# mount -t iso9660 -o loop ../mycd.iso ./iso
fortaleza:/home/matados2k/curso# cd iso
fortaleza:/home/matados2k/curso/iso# ls
autorun.inf   devel                dyne.png      isolinux       logo.png
ChangeLog     dyne                 extras        LICENSE.TXT    README.TXT
CLICK_ME.HTM  dynebolic-manual.pdf  floppy        linuxboot.cfg
default.xbe   dynebol.ico          gnulinux.png  loadlin
fortaleza:/home/matados2k/curso/iso#
```

Despedida.

Espero que esta serie de entregas os haya sido de gran utilidad, y ved la potencia que tienen los sistemas GNU/Linux para manejar todo tipo de particiones. En las siguientes entregas veremos cosas sobre el control y monitorización de procesos. Un saludo.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 28. Monitorización y eliminación de procesos.

> % rm God
> rm: God nonexistent

Introducción.

Ya vimos en las entregas 8 y 9 el control de tareas, ahora aprenderemos más sobre cómo monitorizarlas, saber en qué estado se encuentran, cuáles consumen más y cómo eliminarlas.

Cuando ejecutamos una aplicación, lo que hace el sistema es arrancar un proceso que es el que va a usar cpu y los recursos como la memoria, y que en definitiva, compone la aplicación. Una aplicación puede ejecutar muchos procesos distintos, por lo que podríamos decir para simplificarlo que la aplicación es lo que nosotros vemos y los procesos lo que hace posible la aplicación.

Muchas veces la correspondencia es una aplicación = un proceso, pero no tiene por qué ser así, como por ejemplo grandes servidores como apache que pueden tener varios procesos prestando servicio. Empezaremos como siempre por la consola y terminaremos en las siguientes entregas con las distintas posibilidades gráficas para hacer lo mismo.

Echemos un vistazo.

Lo primero que necesitamos es una herramienta para ver qué procesos están corriendo en nuestro sistema, y para ello qué herramienta mejor que:

ps [opciones]

Este comando informa del estado de los procesos en nuestra máquina, que como casi siempre pasa con otros comandos, tiene más opciones que las que vemos y formas de usarlo, así que pasamos a usarlo sin opciones:

```
matados2k@fortaleza:~$ ps
  PID TTY          TIME CMD
 9951 pts/4    00:00:00 bash
 9955 pts/4    00:00:00 ps
matados2k@fortaleza:~$
```

Vemos que nos muestra información sólo de los procesos propios que se están controlados por algún terminal, y uno es bash, que no es más que el intérprete de comandos, y 'ps', que se ve a sí mismo y se muestra. Vemos también varias columnas, la más importante es PID, que nos muestra el identificador de proceso (este es único), TTY indica en qué terminal se está ejecutando, pts indica que es una consola virtual y el número (tengo unas 5 abiertas ahora mismo), TIME es el tiempo que están consumiendo, y CMD indica a qué ejecutable pertenece y cómo fue invocado.

Pero esto sabe a poco y quiero saber los procesos de los otros usuarios, no solo los míos, y para eso necesito la opción 'a':

```
matados2k@fortaleza:~$ ps a
  PID TTY      STAT   TIME COMMAND
 8014 tty1     Ss+    0:00 /sbin/getty 38400 tty1
 8015 tty2     Ss+    0:00 /sbin/getty 38400 tty2
 8016 tty3     Ss+    0:00 /sbin/getty 38400 tty3
 8017 tty4     Ss+    0:00 /sbin/getty 38400 tty4
 8018 tty5     Ss+    0:00 /sbin/getty 38400 tty5
 8019 tty6     Ss+    0:00 /sbin/getty 38400 tty6
 8713 pts/2    Ss+    0:00 /usr/bin/kdesu_stub -
 9826 pts/3    Ss     0:00 /bin/bash
 9830 pts/3    S+     0:00 man ps
 9837 pts/3    S+     0:00 sh -c /usr/bin/zsoelim /tmp/zmantCBzr4 | /usr/bin/tbl
 9842 pts/3    S+     0:00 /usr/bin/pager -s
10187 pts/5    Ss     0:00 /bin/bash
10442 pts/5    R+     0:00 ps a
matados2k@fortaleza:~$
```

Ya ahora en TTY nos aparece tty, que no es más que los terminales que no están bajo Xque tenemos en nuestro ordenador, y STAT nos indica en qué estado se encuentra el proceso, pudiendo ser los siguientes:

R para preparado para ejecución (runnable)

S para durmiendo (sleeping)

D para indicar letargo ininterrumpible (uninterruptible sleep)

T para parado o trazado (traced)

Z para un proceso zombie.

La letra pequeña de STAT se muestra si hay páginas residentes en memoria, que de momento no nos interesa, y un signo que indica si un proceso tiene prioridad positiva o negativa. Un proceso duerme cuando está esperando algo (vamos a explicarlo todo de forma muy simplificada), si está con D es porque está dormido y no se le puede interrumpir, parado cuando por ejemplo se esta depurando un programa, o zombie cuando ha muerto pero otro proceso tiene que recogerdatos que generó.

Ahora sabemos más cosas, pero no sabemos a quién leches pertenecen esos procesos,y eso no puede ser, opción 'u':

```
matados2k@fortaleza:~$ ps au
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      8014  0.0  0.0   1500   484 tty1     Ss+  17:06   0:00 /sbin/getty 384
root      8015  0.0  0.0   1500   484 tty2     Ss+  17:06   0:00 /sbin/getty 384
root      8016  0.0  0.0   1500   484 tty3     Ss+  17:06   0:00 /sbin/getty 384
root      8017  0.0  0.0   1500   484 tty4     Ss+  17:06   0:00 /sbin/getty 384
root      8018  0.0  0.0   1500   484 tty5     Ss+  17:06   0:00 /sbin/getty 384
root      8019  0.0  0.0   1500   484 tty6     Ss+  17:06   0:00 /sbin/getty 384
root      8713  0.0  0.1   1692   572 pts/2    Ss+  17:19   0:00 /usr/bin/kdesu_
1000      9826  0.0  0.3   3132  1756 pts/3    Ss   17:39   0:00 /bin/bash
1000      9830  0.0  0.2   2436  1408 pts/3    S+   17:39   0:00 man ps
1000      9837  0.0  0.2   2756  1236 pts/3    S+   17:39   0:00 sh -c /usr/bin/
1000      9842  0.0  0.1   1848   652 pts/3    S+   17:39   0:00 /usr/bin/pager
1000     10187  0.0  0.3   3132  1756 pts/5    Ss   17:49   0:00 /bin/bash
1000     10979  0.0  0.1   2784   904 pts/5    R+   18:12   0:00 ps au
matados2k@fortaleza:~$
```

Ya lo tenemos, pero ahora vemos más información: %CPU que nos indica qué porcentaje de procesador está usando, %MEM lo mismo pero con la memoria, VSZ que nada tiene que ver con las siglas de un nuevo deportivo sino del tamaño virtual del proceso en kbytes , RSS el tamaño que ocupa en memoria en kbytes y START la hora en la que comenzó.

Perfecto, nos estamos haciendo unos pequeños expertos, pero seguro que hay algo que se nos escapa, seguro que hay mucho más procesos que no están bajo la supervisión de ningún terminal. Usemos la opción 'x':

```
matados2k@fortaleza:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1504   512 ?        S    17:05   0:00 init [2]
root         2  0.0  0.0      0     0 ?        SN   17:05   0:00 [ksoftirqd/0]
.....
1000     10187  0.0  0.3   3132  1756 pts/5    Ss   17:49   0:00 /bin/bash
1000     11442  0.0  0.0      0     0 ?        Z    18:27   0:00 [pla] <defunct>
1000     11784  0.0  0.1   2784   904 pts/5    R+   18:37   0:00 ps aux
matados2k@fortaleza:~$
```

La hemos acortado porque suele ser larguísima, así que ya podéis hacer uso de las pipes y combinarla con 'more' o 'less' para poderla leer tranquilamente.

La lista de los que más consumen.

Si nuestra intención es saber qué procesos son los que más consumen, el comando 'ps aux' se nos hace muy largo y pesado para ir comprobando uno a uno, para esto tenemos comandos mejores:

top [d intervalo]

Este comando nos muestra de forma interactiva los procesos que másconsumen, refrescando la lista cada 5 segundos normalmente, y tiene opciones que no veremos. Pasemos a ejecutarlo entonces:

```
matados2k@fortaleza:~$ top
top - 18:55:38 up  1:50,  1 user,  load average: 0.14, 0.15, 0.12
Tasks: 102 total,   2 running,  97 sleeping,   1 stopped,   2 zombie
Cpu(s):  4.9% us,  1.0% sy,  0.0% ni, 93.1% id,  0.9% wa,  0.0% hi,  0.0% si
Mem:    516352k total,   491352k used,    25000k free,    29788k buffers
Swap:   530100k total,     2236k used,   527864k free,   212980k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 8027 root      15   0 94596  16m 3436 S  5.9  3.2   2:02.48 XFree86
 9129 matados2  15   0 26864  14m 3828 S  2.0  2.9   0:39.40 wish
    1 root      16   0  1504  512  452 S  0.0  0.1   0:00.47 init
    2 root      34  19     0    0    0 S  0.0  0.0   0:00.10 ksoftirqd/0
    3 root       5 -10     0    0    0 S  0.0  0.0   0:00.16 events/0
    4 root       9 -10     0    0    0 S  0.0  0.0   0:00.01 khelper
   16 root      15 -10     0    0    0 S  0.0  0.0   0:00.00 kacpid
   97 root       5 -10     0    0    0 S  0.0  0.0   0:00.09 kblockd/0
  131 root      15   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  132 root      15   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  134 root      14 -10     0    0    0 S  0.0  0.0   0:00.00 aio/0
  133 root      15   0     0    0    0 S  0.0  0.0   0:00.03 kswapd0
  721 root      25   0     0    0    0 S  0.0  0.0   0:00.00 kseriod
 1023 root      15   0     0    0    0 S  0.0  0.0   0:00.02 kjournald
 1074 root      14  -4  1492  460  400 S  0.0  0.1   0:00.12 udevd
 2728 root      19   0     0    0    0 S  0.0  0.0   0:00.00 kjournald
 2729 root      15   0     0    0    0 S  0.0  0.0   0:00.01 kjournald
matados2k@fortaleza:~$
```

Para salir de 'top' pulsamos la tecla 'q' o Ctrl+C . Vemos ahora nueva información que os dejo como tarea para que la averigüéis vosotros.

Vamos a eliminar.

Ya vimos en las entregas 8 y 9 cómo usar el comando kill, ya sea por PID o el número de tarea. Realmente 'kill' lo que hace es enviar una señal a un proceso y de momento sólo nos interesan las de eliminación, como son la señal número 9 (SIGKILL) y la 15 (SIGTERM). La primera hace que un proceso termine inmediatamente, y la segunda le da oportunidad de terminar ordenadamente, esta ultima puede ser ignorada por el proceso.

Nosotros usaremos kill de la siguiente forma:

kill [-NUMERO|NOMBRE SEÑAL] NºTAREA | PID

Con lo cual podemos especificar la señal, ya sea por su número o su nombre y luego el proceso de 1 de las 2 formas, normalmente usaremos su PID para evitar confusiones, normalmente sacados con 'ps'.

Veamos un simple ejemplo:

```
matados2k@fortaleza:~$ yes > /dev/null&
[1] 10073
matados2k@fortaleza:~$ ps aux | grep yes
1000     10073 95.2  0.0   1784   436 pts/4    R    20:31   0:14 yes
1000     10082  0.0  0.1   2044   764 pts/4    R+   20:32   0:00 grep yes
matados2k@fortaleza:~$ kill -15 10073
matados2k@fortaleza:~$
```

Preferiremos matar antes con 15, y si no respondiera a esa señal usaríamos 9. Como esto ya es algo que estaba visto por encima, lo damos por repasado.

Voy a matarlos a todos.

Imaginemos que una aplicación crea muchos procesos con el mismo nombre, ir PID por PID para eliminarlos puede ser una larga y dura tarea, necesitamos una herramienta para 'matar' en masa y para ello tenemos lo siguiente, usándolo de la siguiente forma:

killall [-NUMERO|NOMBRE SEÑAL] nombre

Vamos a crearnos muchos procesos del maravilloso e inútil comando 'yes', y como son muchos y cobardes y nosotros perezosos e ingeniosos, los eliminaremos de una sola vez:

```
matados2k@fortaleza:~$ yes > /dev/null&
[1] 10539
....
matados2k@fortaleza:~$ yes > /dev/null&
[20] 10561
matados2k@fortaleza:~$ killall -15 yes
matados2k@fortaleza:~$ ps aux | grep yes
1000     10578  0.0  0.1   2044   768 pts/4    S+   20:44   0:00 grep yes
[1]   Terminado              yes >/dev/null
[2]   Terminado              yes >/dev/null
....
[20]+  Terminado               yes >/dev/null
matados2k@fortaleza:~$ ps aux | grep yes
1000     10588  0.0  0.1   2044   764 pts/4    R+   20:45   0:00 grep yes
matados2k@fortaleza:~$
```

Como vemos, nos hemos cargado 20 procesos de 'yes'de forma ordenada, simplemente con su nombre :), somos unos fieros sanguinarios.

Despedida

Pues ya con esto damos por concluida la entrega. En la próxima entrega veremos cómo realizar este tipo de tareas gráficamente.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 29. Monitorización y eliminación de procesos (y II).

```
% man you
No manual entry for you
```

Continuando.

En esta última entrega de la serie de monitorización vamos a ver qué opciones gráficas tenemos para monitorizar y eliminar procesos, como ya dije en la anterior entrega.

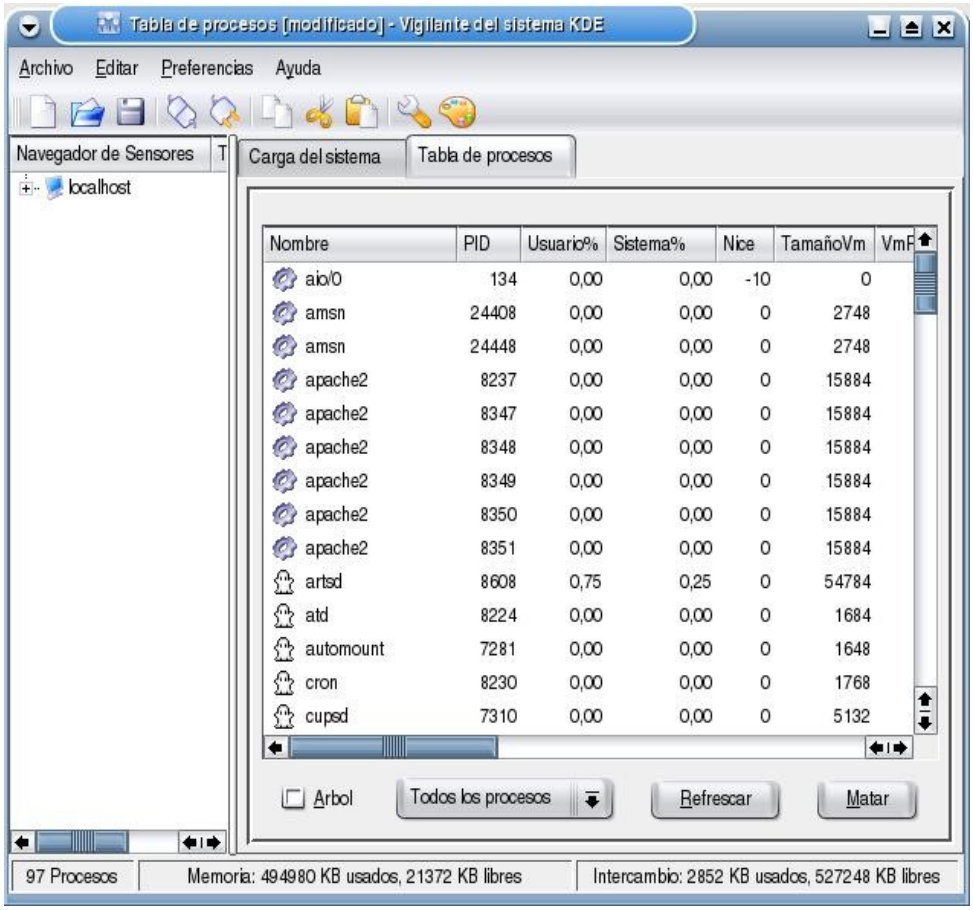
Veamos de qué podemos disponer:

- Guardián del Sistema KDE (ksysguard)
- Monitor del sistema (GNOME) (gnome-system-monitor)

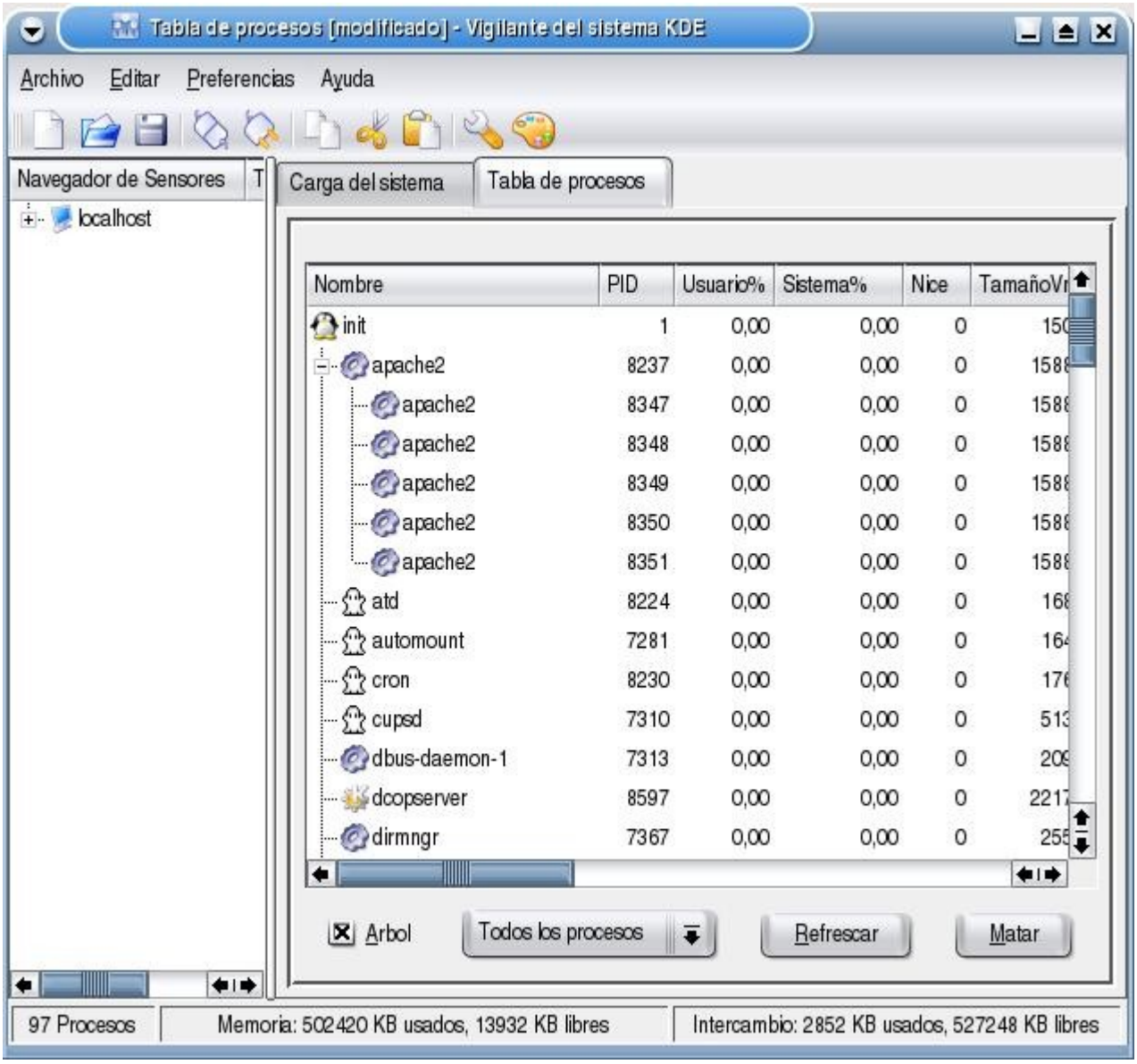
Vamos a ver solamente estas dos, ya que son las de los principales escritorios, si bien es posible usar la de GNOME en KDE y viceversa, incluso desde otro entornos. Podéis buscarlo en el menú de vuestra distribución favorita, pues aunque puede cambiar de unas a otras suelen estar en los menús referentes a la administración o al sistema. En el caso de Debian, que es la que yo uso, las encontramos desde KDE dentro de menú Sistema y en GNOME seguramente estarán en Debian>Aplicaciones>Sistema. En caso de que no las encontréis probad a ejecutarlas en consola con el nombre que os puse entre paréntesis.

Guardián del sistema KDE.

Este es el primero que vamos a ver, y su aspecto es el siguiente:

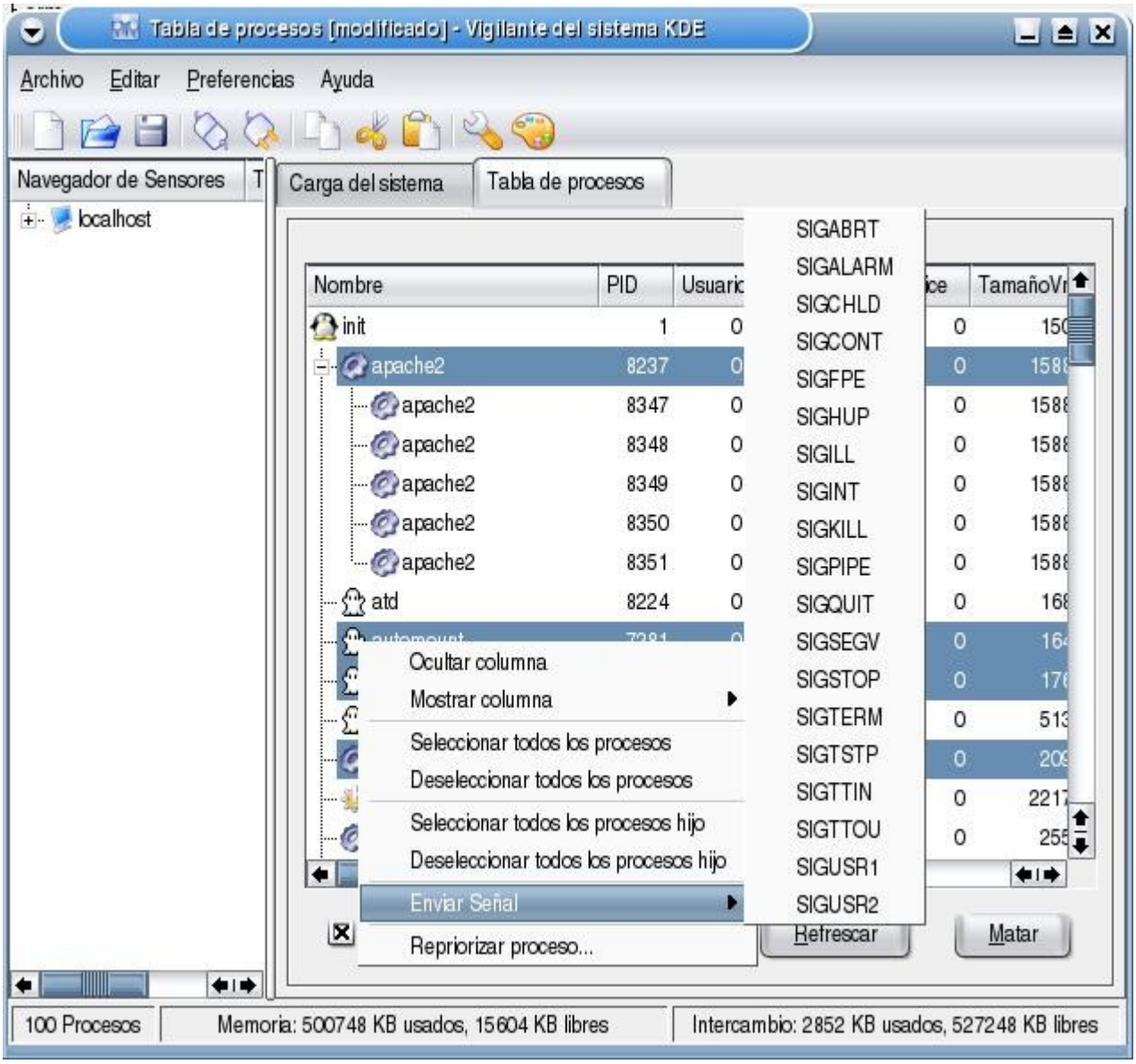


Tenemos en la pestaña “Tabla de procesos” todo lo que conseguimos con el comando 'ps' de la entrega anterior, y además podemos ordenarla por correspondencia padre->hijo, marcando la casilla de árbol.



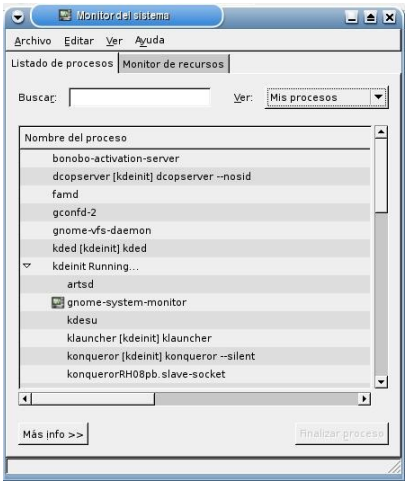
Os acordáis de las opciones de 'ps', 'a','u' y 'x', pues las tenemos en el desplegable de al lado:

Y para eliminar un proceso fácilmente, hacemos una selección como si fueran ficheros y pulsamos en matar, o si quisiéramos mandarle cualquier otra señal seleccionamos ypulsamos el botón derecho del ratón:

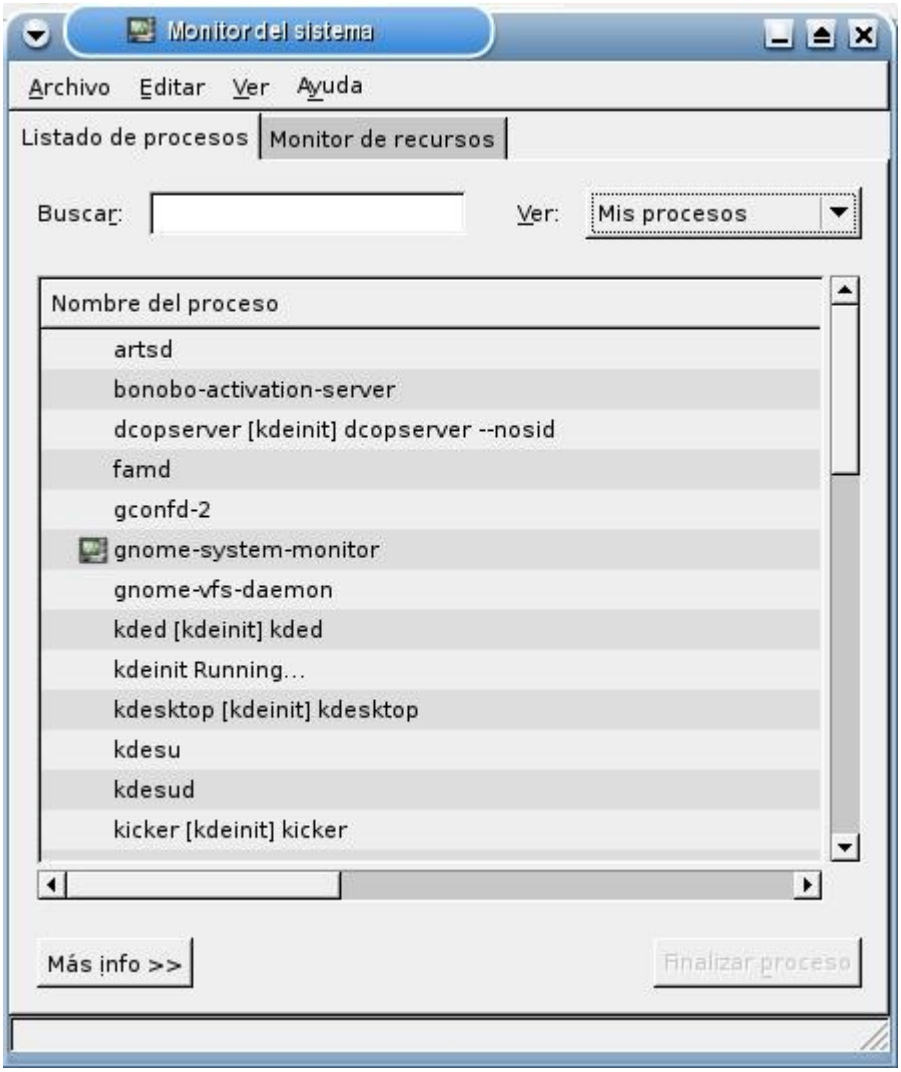


Monitor del sistema.

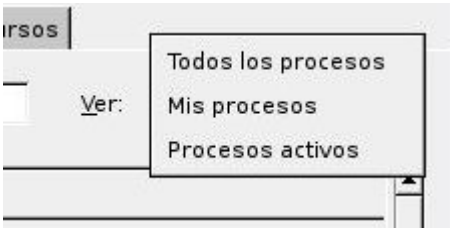
Ahora nos centraremos en nuestra otra opción, también igual de sencilla y útil que la ya vista. Nada más arrancar nos encontramos con lo siguiente:



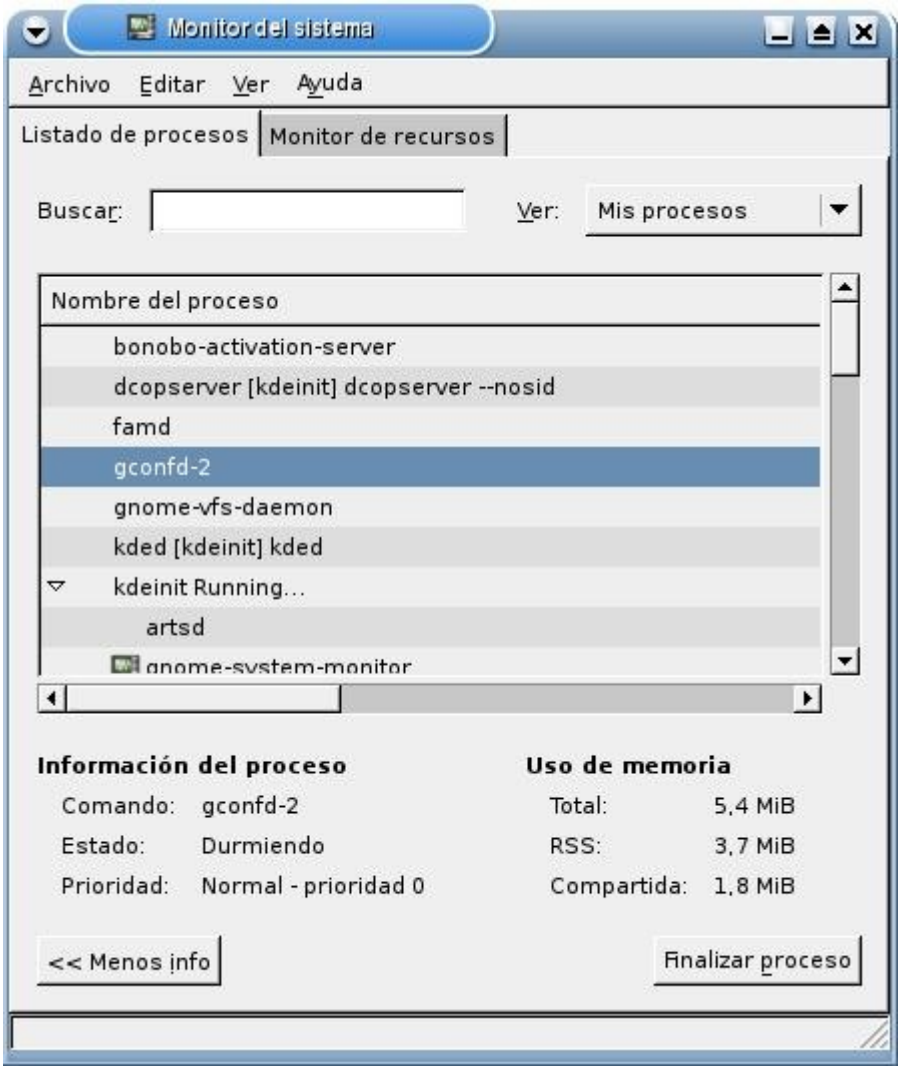
Como observamos, ya por defecto nos sale en forma de árbol, por lo que para quitar esto debemos irnos al menú ver y deseleccionar "Dependencias del proceso":



Las opciones similares al comando 'ps' las tenemos en el botón desplegable de arriba a la derecha:



Al igual que con la aplicación anterior, podemos matar uno o varios procesos realizando una selección, pero en este caso nuestro botón se llama "Finalizar proceso". En este caso no podemos mandarle cualquier otro tipo de señal, y para ver más información de los procesos debemos hacerlo explícitamente activando el botón "Más info" y seleccionando un proceso:



Despedida

Con esto damos por terminada esta sencilla entrega que no implica ninguna dificultad, pero nos da información de las alternativas que tenemos bajo el entorno gráfico. Personalmente a mí me parece mucho más cómoda y útil la primera comentada, pero eso ya es cosa de gustos y de 'usos'. Hasta la próxima entrega.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 30. Compresión y descompresión (I).

% man you
No manual entry for you.

Introducción.

En esta nueva serie de entregas vamos a ver todo lo necesario para comprimir y descomprimir en nuestro sistema GNU/Linux, como de costumbre empezaremos desde la consola de comandos y continuaremos por la interfaz gráfica.

Empaquetando y desempaquetando.

Lo primero que veremos no es propiamente un compresión o descompresión dedatos, sino un empaquetamiento de los mismos. Esto es algo muy común en los sistemas basados en Unix como Linux, primero realizar un empaquetamiento en donde lo único que se hace es pasar varios archivos a uno solo (como el que guarda libros en una caja, o encuaderna varios fascículos y lo convierte en un libro).

Esto lo conseguimos con el comando:

tar [opciones] [nombre_fichero.tar] [ficheros]

Primero empezaremos empaquetando y para eso necesitamos dos opciones: '-c' para decirle que vamos a crear un archivo nuevo y '-f' para decirle que use el sistema de ficheros.

```
matados2k@fortaleza:~$ tar -cf curso.tar curso
matados2k@fortaleza:~$ ls
amsn_received  curso  curso.tar  Desktop  matados2k  matados2k.tar.gz
matados2k@fortaleza:~$
```

Ahora necesitamos realizar el paso contrario y para ello disponemos de las opciones '-x' para decirle que queremos extraer, '-v' para que nos diga qué es lo que va haciendo y 'f' para lo mismo comentado anteriormente:

```
matados2k@fortaleza:~$ mv curso cursoOLD
matados2k@fortaleza:~$ ls
amsn_received  cursoOLD  curso.tar  Desktop  matados2k  matados2k.tar.gz
matados2k@fortaleza:~$ tar -xvf curso.tar
curso/
curso/supertux-0.1.2/
......
curso/getleft_1.1.2-2_all.deb
curso/getleft_1.1.1-2_all.deb
matados2k@fortaleza:~$ ls
amsn_received  curso  cursoOLD  curso.tar  Desktop  matados2k  matados2k.tar.gz
matados2k@fortaleza:~$
```

Pero... ¿y si nos interesa ver el contenido del '.tar' antes de descomprimir? Pues usamos la opción '-t' con '-f', lo que hará que nos muestre todo el contenido dentro del fichero '.tar' de la misma forma que lo vimos al descomprimir.

Como vemos, este comando puede ser una buena opción para crear nuestras copias de seguridad, pero tiene una pega y es que no comprime, cosa que solucionaremos más adelante.

Compresión y descompresión en gzip.

Gzip, con extensión '.gz', es uno de los formatos más comunes que podemos encontrar dentro del mundo del software libre (formato de compresión LZ77), en un principio mantenido por Jean-loup Gailly y actualmente por la Free Software Fundation, Inc. Empecemos viendo cómo usarlo:

gzip [-opciones] [ficheros]

Y lo primero es comprimir un fichero, para ello usaremos gzip sin opción:

```
matados2k@fortaleza:~$ gzip curso.tar
matados2k@fortaleza:~$ ls
amsn_received  cursoOLD      Desktop    matados2k.tar.gz
curso          curso.tar.gz  matados2k
matados2k@fortaleza:~$
```

Como observaréis no ha hecho falta indicar una extensión, nuestro fichero resultante es curso.tar.gz. Pues bien, acabamos de crear un fichero con una de las extensiones que más os encontrareis: 'tar.gz', y ahora ya sabéis su significado. Ojo, daos cuenta de que 'curso.tar' ha sido sustituido por 'curso.tar.gz'. Para descomprimirlo usamos la opción '-d' y el nombre del fichero:

```
matados2k@fortaleza:~$ gzip -d curso.tar.gz
matados2k@fortaleza:~$ ls
amsn_received  curso  cursoOLD  curso.tar  Desktop  mata
matados2k@fortaleza:~$
```

Ahora pasamos de tener un archivo .tar.gz a un .tar si os fijáis. Para ver el contenido de un .tar.gz usaremos la opción '-l'.

Pues yo lo quiero todo en uno.

Sí, lo sé, es muy pesado usar dos comandos para una cosa simple, pero esto tiene fácil solución: el comando 'tar' es capaz de hacerlo directamente usando la opción '-z' en cada una de las formas que vimos antes (incluso para ver su contenido):

```
matados2k@fortaleza:~$ tar -czf curso.tar.gz curso
matados2k@fortaleza:~$ ls
amsn_received  cursoOLD   curso.tar.gz  matados2k
curso          curso.tar  Desktop       matados2k.tar.gz
matados2k@fortaleza:~$ matados2k@fortaleza:~$ tar -xvzf curso.tar.gz
curso/
curso/supertux-0.1.2/
....
curso/getleft_1.1.2-2_all.deb
curso/getleft_1.1.1-2_all.deb
matados2k@fortaleza:~$
```

Quiero una mejor compresión, bzip2.

Este formato de compresión está basado en el algoritmo de compresión Burrows-Wheeler block sorting text y Huffman coding, que suele comportarse mejor que el anterior y su autor es Julian Seward. Y por suerte su uso es igual que gzip (para lo que nosotros lo hemos usado, otras opciones pueden cambiar) así que veamos la diferencia entre uno y otro:

```
matados2k@fortaleza:~$ bzip2 curso.tar
matados2k@fortaleza:~$ ls -l
total 3599112
drwx------   2 matados2k matados2k       4096 2005-06-01 20:29 amsn_received
drwxr-xr-x   4 matados2k matados2k       4096 2005-05-16 08:40 curso
drwxr-xr-x   4 matados2k matados2k       4096 2005-05-16 08:40 cursoOLD
-rw-r--r--   1 matados2k matados2k   31261885 2005-06-12 18:24 curso.tar.bz2
-rw-r--r--   1 matados2k matados2k   33918144 2005-06-14 19:18 curso.tar.gz
drwxr-xr-x   5 matados2k matados2k       4096 2005-06-07 23:26 Desktop
drwxr-xr-x  32 matados2k matados2k       4096 2005-06-14 19:09 matados2k
-rwxrwxrwx   1 matados2k matados2k 3616666651 2005-05-08 23:44 matados2k.tar.gz
matados2k@fortaleza:~$
```

Como veis hemos comprimido más, la pega ... que tarda también bastante más, así que sopesad vosotros mismo si preferís tiempo o espacio ;)

En este caso el comando 'tar' ya no comprime directamente en este formato así que debemos usar las

tuberías para este menester, ¿que no sabes cómo? Si es así es que no has aprendido con elcurso.

Pues yo quiero el zip de toda la vida.

Para ello utilizaremos dos comandos que lo mismopuede que necesitemos instalar, ellos son 'zip' y 'unzip', y su forma de uso es la siguiente:

zip [-r] nombre_archivo.zip lista_de_ficheros

unzip [-v] nombre_archivo.zip

El primero es para comprimir con la opción '-r' si queremos que nos incluya todo el contenido de un directorio (y no sólo el directorio), y el segundo para descomprimir con la opción '-v' en el caso de que queramos ver sólo el contenido:

```
matados2k@fortaleza:~$ zip -r curso.zip curso
 ...
  adding: curso/getleft_1.1.2-2_all.deb (deflated 0%)
  adding: curso/getleft_1.1.1-2_all.deb (deflated 0%)
matados2k@fortaleza:~$ ls -l
total 3632588
drwx------   2 matados2k matados2k       4096 2005-06-01 20:29 amsn_received
drwxr-xr-x   4 matados2k matados2k       4096 2005-05-16 08:40 curso
drwxr-xr-x   4 matados2k matados2k       4096 2005-05-16 08:40 cursoOLD
-rw-r--r--   1 matados2k matados2k   31261885 2005-06-12 18:24 curso.tar.bz2
-rw-r--r--   1 matados2k matados2k   33918144 2005-06-14 19:18 curso.tar.gz
-rw-r--r--   1 matados2k matados2k   34236271 2005-06-14 19:36 curso.zip
drwxr-xr-x   5 matados2k matados2k       4096 2005-06-07 23:26 Desktop
drwxr-xr-x  32 matados2k matados2k       4096 2005-06-14 19:09 matados2k
-rwxrwxrwx   1 matados2k matados2k 3616666651 2005-05-08 23:44 matados2k.tar.gz
matados2k@fortaleza:~$
```

Como vemos, en este caso gzip y bzip2 ganan.

Despedida

En la próxima entrega veremos los formatos referentes a lha, arj, zoo y rar. Un saludo y hasta la próxima.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 31. Compresión y descompresión (II).

```
% cat catfood
cat: cannot open catfood
```

Ya estamos de nuevo.

Para lo nuevos esta será una entrega más, pero para los viejos seguidores, esta es la entrega que debió salir hace mucho tiempo, realmente mucho, así que aceptaré cualquier tirón de orejas que queráis hacerme llegar. Sin más, damos comienzo a una nueva época dentro del curso.

Un compresor de uso gratuito, pero no libre.

LHA es una utilidad de compresión gratuita y creada en 1988 por Haruyasu Yoshizaki. LHA es un formato muy famoso en Japón y ha sido usado como compresor de ficheros de instalación de juegos tan famosos como “Doom”, por poner un ejemplo.

Veamos su uso:

lha [opciones] archivo{.lhz|.lha} [ficheros]

Lo primero que vamos a realizar es la compresión de un directorio, ypara eso necesitamos la opción 'a', veámoslo:

```
matados2k@imperio:~/curso$ lha a austin_powers.lhz austin_powers.txt
austin_powers.txt       - Frozen(86%) o
matados2k@imperio:~/curso$
```

En el caso de haber comprimido una gran cantidad de ficheros en un mismo fichero, puede que nos sea interesante saber su contenido, pasa eso tenemos la opción 'l' de listar o 'v' de verbose:

```
matados2k@imperio:~/curso$ lha v austin_powers.lhz
 PERMSSN    UID  GID    PACKED    SIZE  RATIO METHOD CRC     STAMP          NAME
---------- ----------- ------- ------- ------ ---------- ------------
-------------
-rw-r--r--  1000/1000       68      79  86.1% -lh7- efa3 Aug  4  2004
austin_powers.txt
---------- ----------- ------- ------- ------ ---------- ------------
-------------
 Total         1 file       68      79  86.1%            Dec 10 20:35
matados2k@imperio:~/curso$ lha l austin_powers.lhz
 PERMSSN    UID  GID      SIZE  RATIO     STAMP           NAME
---------- ----------- ------- ------ ------------ --------------------
-rw-r--r--  1000/1000       79  86.1% Aug  4  2004 austin_powers.txt
---------- ----------- ------- ------ ------------ --------------------
```

```
 Total         1 file       79  86.1% Dec 10 20:35
matados2k@imperio:~/curso$
```

Como podéis observar, la diferencia reside en la cantidad de informaciónmostrada, de la cual la más interesante quizás sea el ratio de compresión.

Pero nosotros necesitamos recuperar nuestra información, necesitamos descomprimiry para ello usamos la opción 'x':

```
matados2k@imperio:~/curso$ lha x austin_powers.lhz
austin_powers.txt OverWrite ?(Yes/[No]/All/Skip) y
austin_powers.txt       - Melted   :  o
matados2k@imperio:~/curso$
```

Como podéis ver, el comando es considerado y nos pregunta antes de sobreescribir nuestros datos, en el caso de que sólo quisiéramos un fichero en concreto bastaría con poner sus nombres a continuación:

```
matados2k@imperio:~/curso$ lha x austin_powers.lhz austin_powers.txt
austin_powers.txt OverWrite ?(Yes/[No]/All/Skip) y
austin_powers.txt       - Melted   :  o
matados2k@imperio:~/curso$
```

Todo un veterano.

Fue inventado por Robert K. Jung. ARJ probablemente significa **A**rchiver **R**obert **J**ung, ya apenas se usa, pero tuvo su gran momento en la época en que los disquetes eran realmente útiles y las copias de seguridad eran M2M, de mano en mano ;). La compresión de ARJ es similar en cierta medida a la de PKZIP 1.02. Algunas partes de ARJ estaban cubiertas por una patente americana, y hoy en día ARJ ha perdido mucha de su cuota de mercado debido a RAR y otros formatos; la falta de una interfaz gráfica también ha contribuido a su desaparición virtual del mundo del escritorio.

El uso es exactamente igual al de lha, por lo que no necesita explicación:

```
matados2k@imperio:~/curso$ arj a austin_powers.arj austin_powers.txt
ARJ32 v 3.10, Copyright (c) 1998-2004, ARJ Software Russia. [24 Nov 2005]

Updating archive  : austin_powers.arj
Archive created: 2006-12-10 22:47:44, modified: 2006-12-10 22:47:44
Replacing austin_powers.txt            86.1%
     1 file(s)
matados2k@imperio:~/curso$ arj l austin_powers.arj
ARJ32 v 3.10, Copyright (c) 1998-2004, ARJ Software Russia. [24 Nov 2005]

Processing archive: austin_powers.arj
```

```
Archive created: 2006-12-10 22:47:44, modified: 2006-12-10 22:47:44
Filename       Original Compressed Ratio DateTime modified Attributes/GUA BPMGS
------------ ---------- ---------- ----- ----------------- -------------- -----
austin_powers.txt
                     79         68 0.861 04-08-04 13:10:06 -rw-r--r-- ---   1
------------ ---------- ---------- -----
     1 files         79         68 0.861
matados2k@imperio:~/curso$ arj v austin_powers.arj
ARJ32 v 3.10, Copyright (c) 1998-2004, ARJ Software Russia. [24 Nov 2005]

Processing archive: austin_powers.arj
Archive created: 2006-12-10 22:47:44, modified: 2006-12-10 22:47:44
Sequence/Pathname/Comment/Chapters
Rev/Host OS    Original Compressed Ratio DateTime modified Attributes/GUA BPMGS
------------ ---------- ---------- ----- ----------------- -------------- -----
001) austin_powers.txt
 11 UNIX             79         68 0.861 04-08-04 13:10:06 -rw-r--r-- ---   1
                                    DTA   04-08-04 13:10:06
                                    DTC   06-12-10 20:52:58
------------ ---------- ---------- -----
     1 files         79         68 0.861
matados2k@imperio:~/curso$ arj x austin_powers.arj
ARJ32 v 3.10, Copyright (c) 1998-2004, ARJ Software Russia. [24 Nov 2005]

Processing archive: austin_powers.arj
Archive created: 2006-12-10 22:47:44, modified: 2006-12-10 22:47:44
ARJ         79 04-08-04 13:10:06, DISK         79 04-08-04 13:10:06
austin_powers.txt          is same or newer, Overwrite? y
Extracting austin_powers.txt           OK
     1 file(s)
matados2k@imperio:~/curso$
```

Pero en este caso sí que hay algo más, y es un comando especifico para descomprimir, aunque ya habéis visto que para nada es necesario:

unarj [opciones] archivo [ficheros]

Veamos su uso:

```
matados2k@imperio:~/curso$ unarj austin_powers.arj
ARJ32 v 3.10, Copyright (c) 1998-2004, ARJ Software Russia. [24 Nov 2005]
```

```
Processing archive: austin_powers.arj
Archive created: 2006-12-10 22:47:44, modified: 2006-12-10 22:49:15
Filename       Original Compressed Ratio DateTime modified Attributes/GUA BPMGS
------------ ---------- ---------- ----- ----------------- ------------- -----
austin_powers.txt
                     79         68 0.861 04-08-04 13:10:06 -rw-r--r-- ---   1
------------ ---------- ---------- -----
     1 files         79         68 0.861
matados2k@imperio:~/curso$
```

Vamos de visita al Zoo.

No, no estoy de broma, así se llama el que toca ahora. Zoo es usado para crear y mantener colecciones de ficheros en forma comprimida, como todos los demás. Usa un algoritmo de compresión Lempel-Ziv que consigue unos ratios de compresión del 20% al 80% dependiendo del tipo de datos. Y cómo no, lo podemos usar exactamente igual que el lha:

```
matados2k@imperio:~/curso$ zoo a austin_powers.zoo austin_powers.txt
Zoo:  austin_powers.txt --  ( 4%) added
matados2k@imperio:~/curso$ zoo l austin_powers.zoo

Archive austin_powers.zoo:
Length    CF  Size Now  Date      Time
--------  --- --------  --------- --------
      79   4%       76   4 Aug 04 13:10:06+64   austin_powers.txt
--------  --- --------  --------- --------
      79   4%       76     1 file
matados2k@imperio:~/curso$ zoo v austin_powers.zoo

Archive austin_powers.zoo:
Length    CF  Size Now  Date      Time
--------  --- --------  --------- --------
      79   4%       76   4 Aug 04 13:10:06+64   austin_powers.txt
--------  --- --------  --------- --------
      79   4%       76     1 file
matados2k@imperio:~/curso$ zoo x austin_powers.zoo
Zoo:  austin_powers.txt -- skipped
matados2k@imperio:~/curso$ zoo x austin_powers.zoo austin_powers.txt
Zoo:  austin_powers.txt -- skipped
matados2k@imperio:~/curso$
```

Ya veis, rápido, sencillo e indoloro, todos iguales :)

Uno de los grandes, RAR.

Y por último, un gran superviviente de la época, aparte del archiconocido zip, que a día de hoy sigue siendo uno de los formatos más utilizados. El formato RAR fue desarrollado por Eugene Roshal y lleva su nombre. RAR significa **R**oshal **Ar**chive.

La primera versión comercial de RAR se lanzó a finales de 1993. Esta primera versión demostró ser más eficaz que la proporcionada por ZIP, por lo que rápidamente se convirtió en el primer competidor de ZIP siendo hoy en día más utilizado, sobre todo para las descargas desde internet. El RAR es más lento que el ZIP, pero comprime mejor y tiene un mayor sistema de redundancia de datos para prevenir errores. RAR utiliza un algoritmo de compresión basado en el LZSS.

A que lo adiviná is, ¡se usa igual que los otros! Y además, al igual que 'arj' tiene 'unarj', 'rar' tiene 'unrar':

```
matados2k@imperio:~/curso$ rar a austin_powers.rar austin_powers.txt

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help

Evaluation copy. Please register.

Creating archive austin_powers.rar

Adding    austin_powers.txt                                         OK
Done
matados2k@imperio:~/curso$ rar l austin_powers.rar

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help

Archive austin_powers.rar

 Name             Size   Packed Ratio  Date   Time     Attr      CRC   Meth Ver
-------------------------------------------------------------------------------
 austin_powers.txt       79       79 100% 04-08-04 13:10 -rw-r--r-- 6062A8E6 m3b
2.9
-------------------------------------------------------------------------------
    1               79       79 100%

matados2k@imperio:~/curso$ rar v austin_powers.rar

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help
```

```
Archive austin_powers.rar

Pathname/Comment
                  Size   Packed Ratio  Date   Time     Attr      CRC   Meth Ver
-------------------------------------------------------------------------------
 austin_powers.txt
                    79       79 100% 04-08-04 13:10 -rw-r--r-- 6062A8E6 m3b 2.9
-------------------------------------------------------------------------------
    1               79       79 100%

matados2k@imperio:~/curso$ rar x austin_powers.rar

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help


Extracting from austin_powers.rar


austin_powers.txt already exists. Overwrite it ?
[Y]es, [N]o, [A]ll, n[E]ver, [R]ename, [Q]uit y

Extracting  austin_powers.txt                                         OK
All OK
matados2k@imperio:~/curso$matados2k@imperio:~/curso$ rar a austin_powers.rar
austin_powers.txt

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help

Evaluation copy. Please register.

Creating archive austin_powers.rar

Adding    austin_powers.txt                                           OK
Done
matados2k@imperio:~/curso$ rar l austin_powers.rar

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help
```

```
Archive austin_powers.rar

 Name             Size   Packed Ratio  Date   Time     Attr      CRC   Meth Ver
-------------------------------------------------------------------------------
 austin_powers.txt       79       79 100% 04-08-04 13:10 -rw-r--r-- 6062A8E6 m3b
2.9
-------------------------------------------------------------------------------
    1               79       79 100%


matados2k@imperio:~/curso$ rar v austin_powers.rar
```

```
RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help

Archive austin_powers.rar

Pathname/Comment
                  Size   Packed Ratio  Date   Time     Attr      CRC   Meth Ver
-------------------------------------------------------------------------------
 austin_powers.txt
                    79       79 100% 04-08-04 13:10 -rw-r--r-- 6062A8E6 m3b 2.9
-------------------------------------------------------------------------------
    1               79       79 100%

matados2k@imperio:~/curso$ rar x austin_powers.rar

RAR 3.51   Copyright (c) 1993-2005 Alexander Roshal   7 Oct 2005
Shareware version         Type RAR -? for help


Extracting from austin_powers.rar


austin_powers.txt already exists. Overwrite it ?
[Y]es, [N]o, [A]ll, n[E]ver, [R]ename, [Q]uit y

Extracting  austin_powers.txt                                         OK
All OK
matados2k@imperio:~/curso$ unrar x austin_powers.rar

UNRAR 3.51 freeware      Copyright (c) 1993-2005 Alexander Roshal


Extracting from austin_powers.rar


austin_powers.txt already exists. Overwrite it ?
[Y]es, [N]o, [A]ll, n[E]ver, [R]ename, [Q]uit y

Extracting  austin_powers.txt                                         OK
All OK
matados2k@imperio:~/curso$
```

Como veis, de libre no tiene nada, pero lo importante es poder utilizarlo dado su amplio uso.

Despedida

En la próxima entrega veremos los compresores gráficos. Un saludo y hasta la próxima.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 32. Compresión y descompresión (y III).

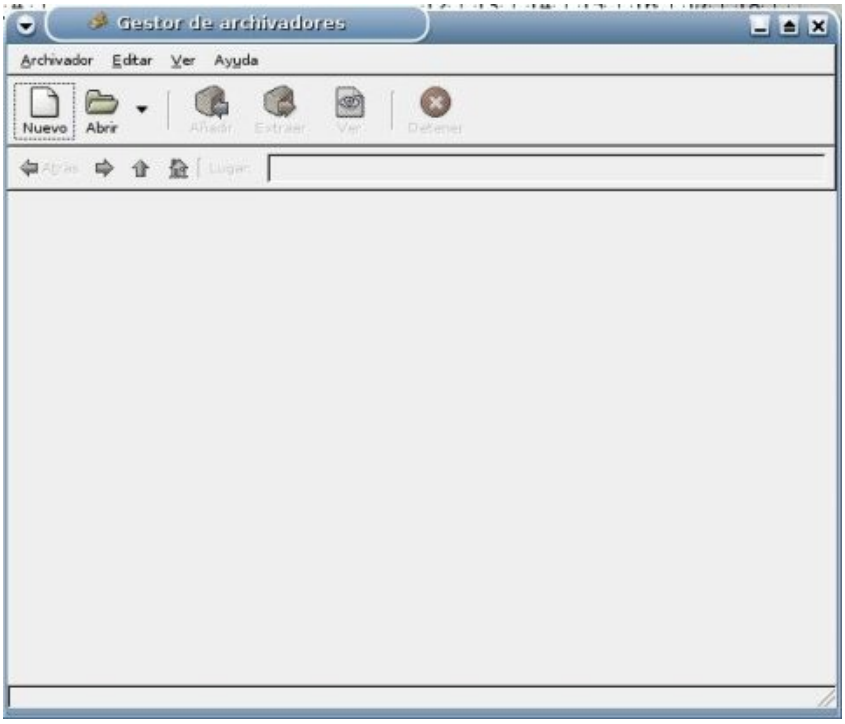
```
% cat catfood
cat: cannot open catfood
```

Y ya termina la serie interminable.

Y digo la serie interminable porque con compresión y descompresión fue donde lo dejé en la anterior versión del curso, que de eso hace ya mucho tiempo. Para terminar veremos dos gestores gráficos, el primero es 'File-roller' de GNOME y el segundo 'Ark' de KDE.

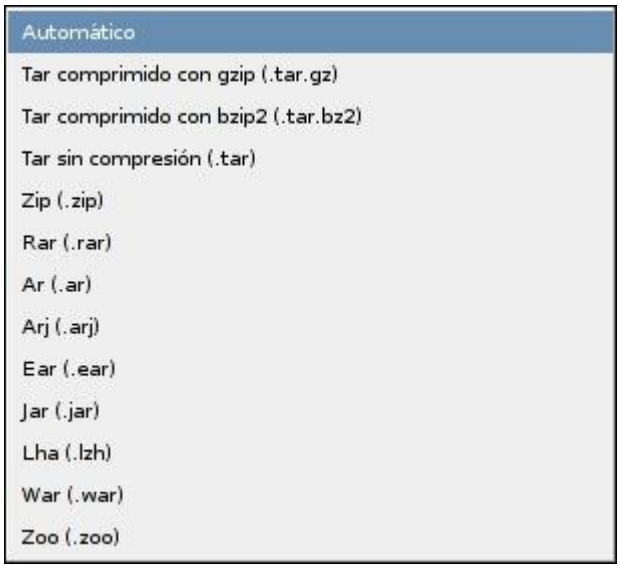
File-roller.

Una vez arrancamos la aplicación la vemos tal que así:

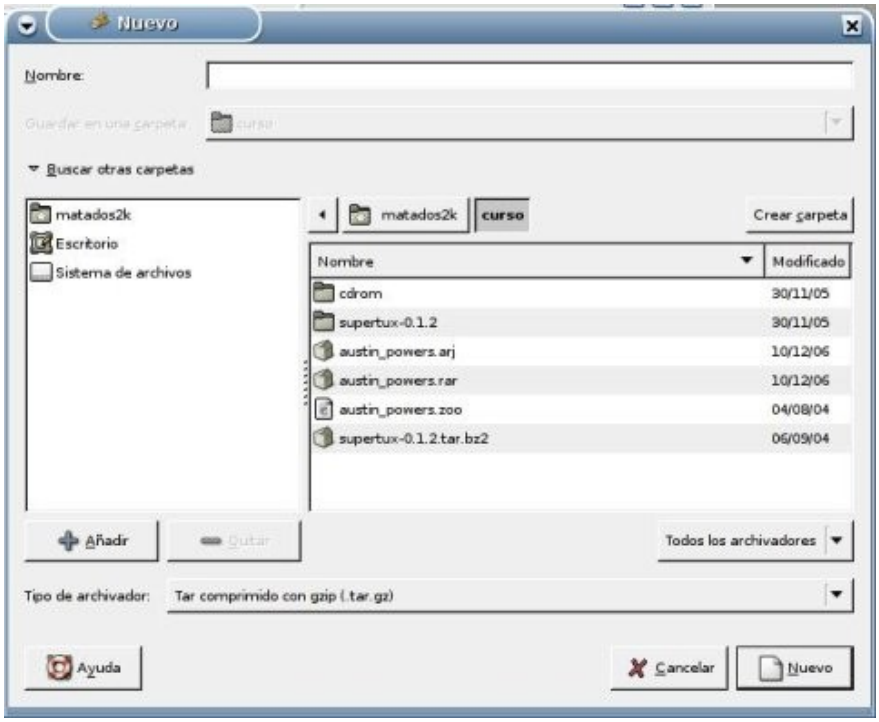


Su uso además es bien sencillo, para crear nuestro archivo pulsamos en el inmenso botón de “Nuevo”:

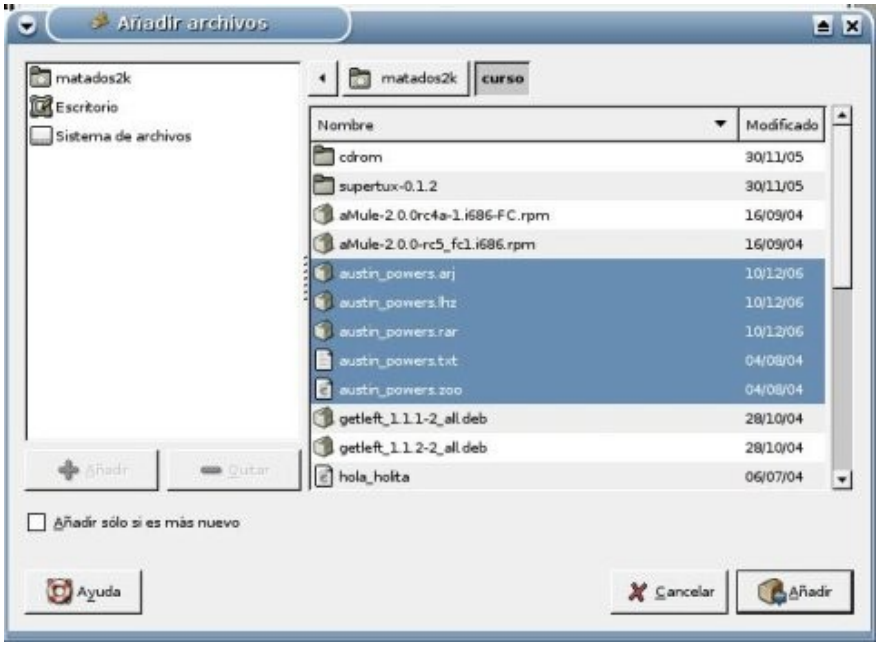
Donde pone “Tipo de archivador”, pulsamos para que se nos despliegue una lista donde elegir el tipo de archivo a crear, que no es más que los compresores que tenemos instalados en nuestra máquina:



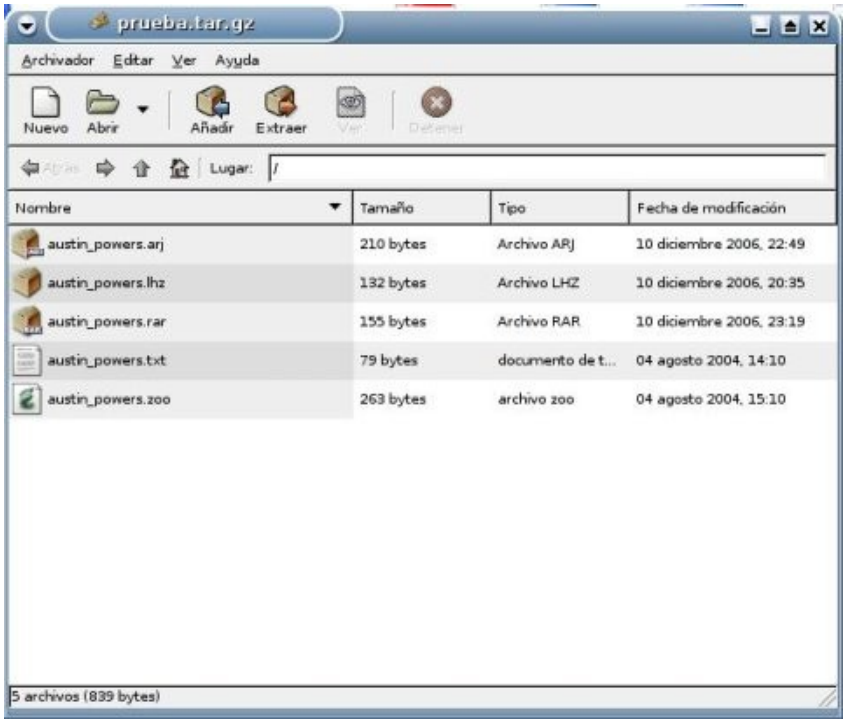
Una vez elegido, desplegamos la parte de “Buscar en otras carpetas” para crear nuestro archivo exactamente donde necesitamos:



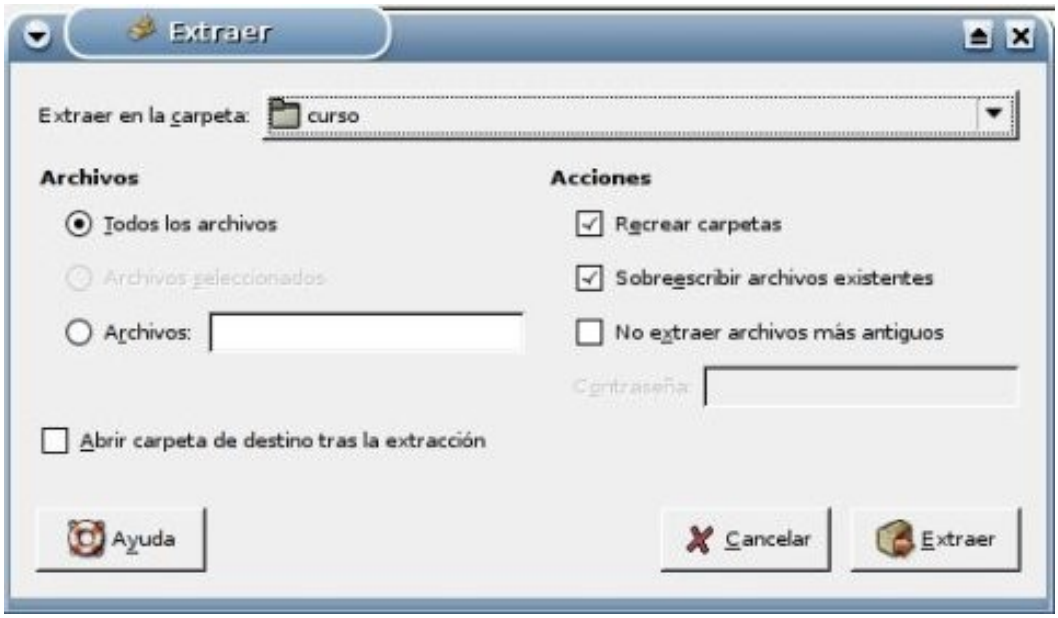
Le damos el nombre que queremos y pulsamos en “Nuevo”. A continuación vamos a añadir los archivos que queremos a nuestro comprimido y para ello pulsamos sobre el botón 'Añadir' que tenemos activos, donde podremos hacer una selección múltiple de lo que deseamos:



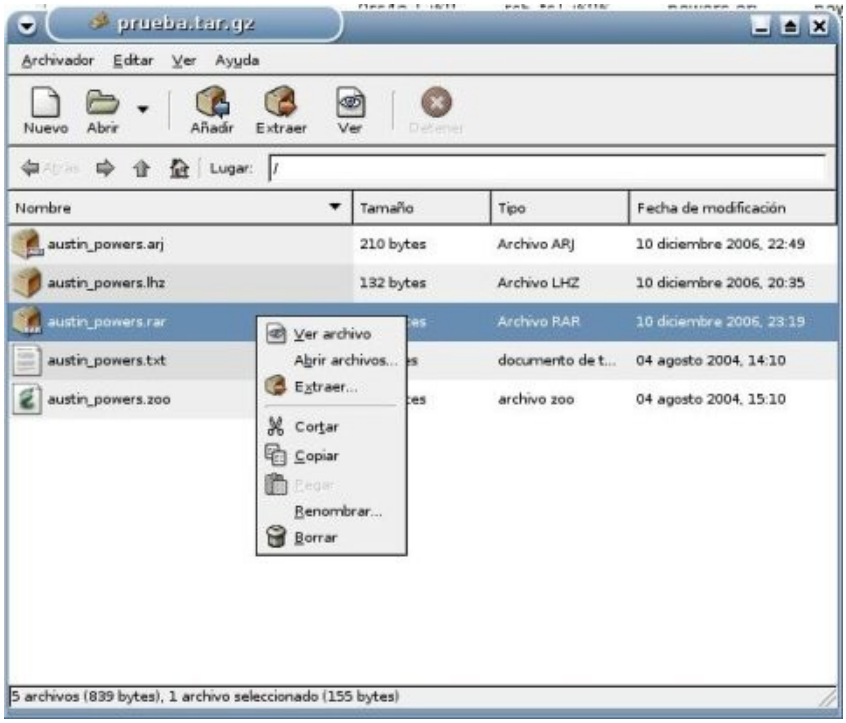
Una vez terminemos, pulsando en “Añadir” conseguiremos nuestro fichero comprimido deseado:



Cada vez que abramos el archivo lo veremos de esta forma, que es el listado de lo que contiene, para extraer el contenido o bien hacemos una selección de lo que queremos y luego pulsamos "Extraer" o bien no seleccionamos nada y lo extraemos todo pulsando en "Extraer":

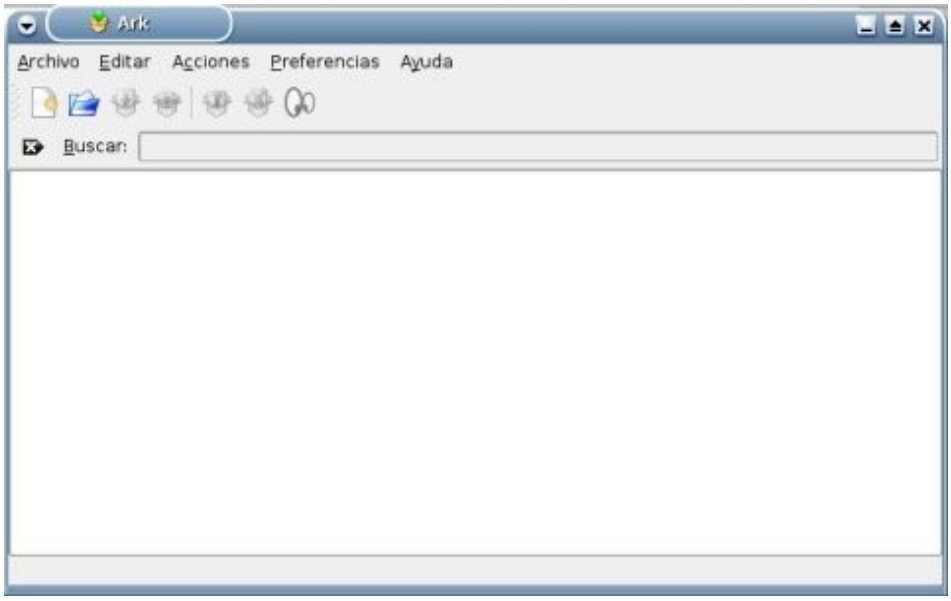


Para visualizar un archivo sin extraerlo disponemos del botón "Ver". Podemos manejar los elementos individualmente pulsando el botón derecho del ratón para un menú contextual que nos permitirá hacer las cosas más necesarias, más fácil imposible.



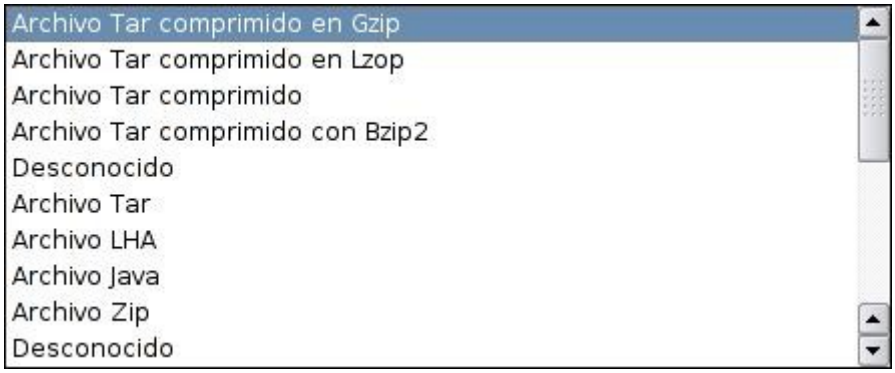
Ark.

Como vamos a ver, "Ark" es igual de simple y se usa de idéntica forma, nada más abrirlo se nos presenta así:



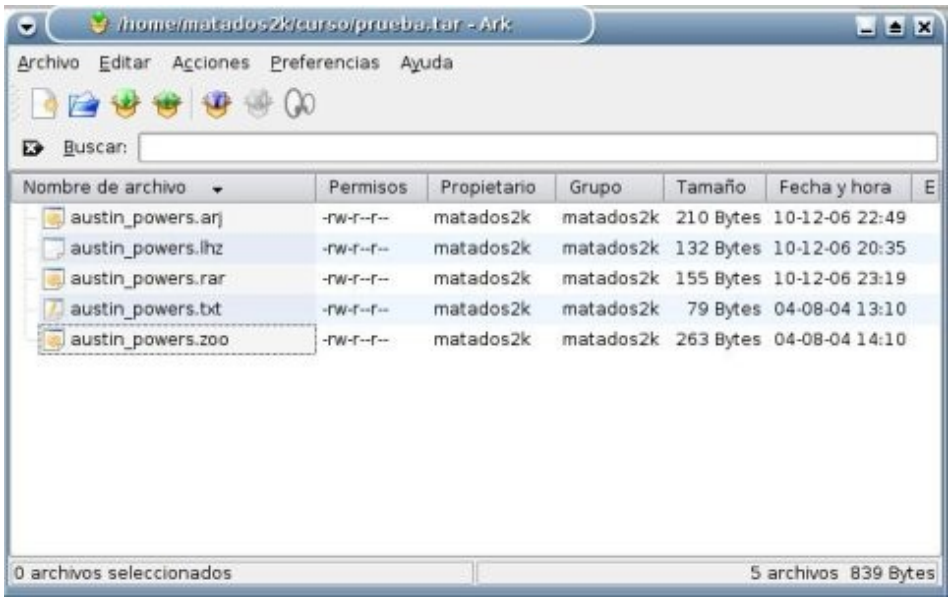
Para crear nuestro archivo pulsamos sobre "Nuevo", que es el primer botón, y se nos abre la ventana de selección de destino y creación de fichero:

En esta ocasión, en la parte del filtro se nos deja también elegir el tipo de fichero, este es aún más completo que el anterior:

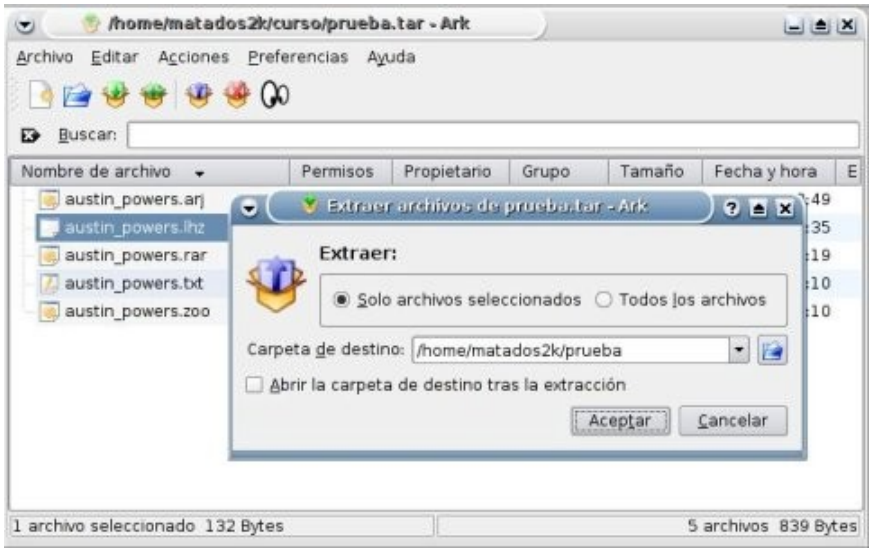


En mi caso elijo Tar comprimido con Bzip2, y una vez creado pulso sobre "Añadir archivos" que es el tercer botón, aunque podría haber usado 'Añadir carpetas' que es el cuarto:

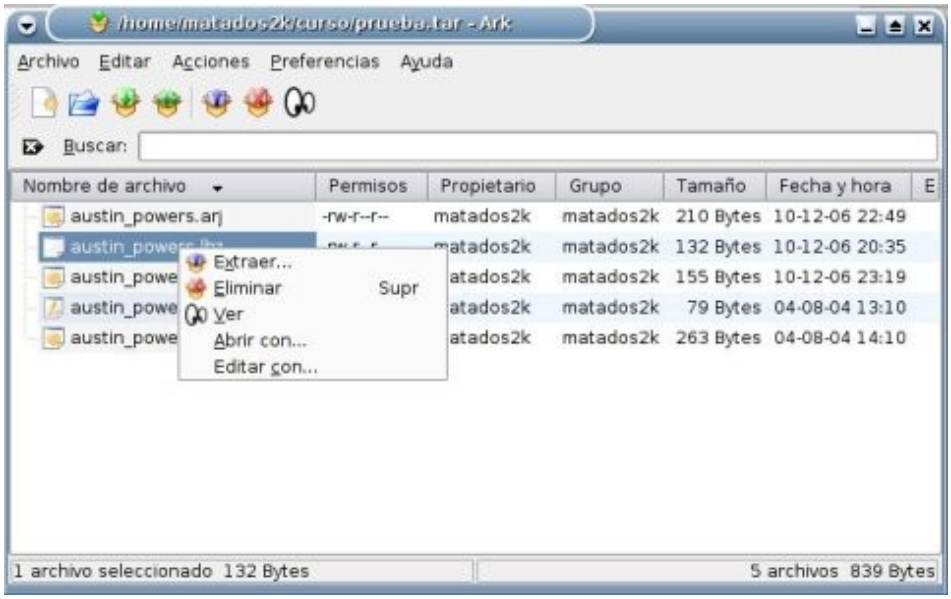
Una vez hecha la selección múltiple de lo que quiero añadir, el resultado es el siguiente:



Cada vez que abramos nuestro archivo comprimido nos saldrá esta ventana y para poder descomprimir o bien seleccionamos los archivos y pulsamos en "Extraer" que es el quinto botón, o bien no seleccionamos ninguno y pulsamos en "Extraer" para extraerlos todos:



Y como en el anterior, para visualizar un archivo sin extraerlo disponemos del botón "Ver". Podemos manejar los elementos individualmente pulsando el botón derecho del ratón para un menú contextual que nos permitirá hacer las cosas más necesarias, igual que antes.. más fácil imposible.



Despedida.

Como veis, la entrega de esta semana no tiene ninguna dificultad y es bien sencilla, pero habría que ver todas las alternativas. La siguiente entrega comenzará con el arranque y parada en Linux, cosa que es bastante más interesante. Hasta la próxima semana.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 33. Administración de servicios (I).

```
% How's my lovemaking?
Unmatched '.
```

Introducción.

Vamos a ver en esta serie de entregas cómo se administran los servicios en Linux. Un servicio es una función específica que realiza la máquina, como un servidor de base de datos, un servidor http o un servidor ftp.

Estos servicios están atendidos por demonios. Un demonio es un proceso en segundo plano que está a la espera de peticiones para realizar una tarea. También se les suele llamar deamon. Según la wikipedia, el origen de la palabra daemon (demonio) se encuentra en la antigua Grecia y la figura del daimon, un espíritu interior, equivalente a un "ángel protector" que guiaba y protegía a los hombres, pero según otras fuentes la palabra daemon viene de los “Day-monitor” de MULTICS. Comenzaron a llamarse en UNIX “daymons”, y en la zona de EEUU donde se trabajaba con UNIX se pronunciaba de la misma forma que “daemon”.

Un demonio puede atender varios servicios, e incluso puede darse el caso que un servicio es dado por varios demonios, pero lo normal va a ser siempre un demonio por servicio.

Pero, podéis pensar que si no vais a destinar vuestra máquina a ningún tipo de servidor, ¿para qué me sirven estas entregas? Es fácil, los servicios no sólo son para servidores como he puesto como ejemplo anteriormente, hay multitud de servicios como “samba” para las redes de Microsoft, los de sonido para que escuches tu música, cups para que puedas imprimir y muchísimos más.

No nos centraremos en cómo se configuran cada uno de ellos en esta serie, sino que vamos a aprender cómo arrancarlos, pararlos, reiniciarlos, que arranquen automáticamente, que nunca arranquen, etc...

Esto, por si aún no le has encontrado la utilidad inmediata, te puede servir para optimizar los recursos de tu sistema, ya que un servicio que no se usa es un gasto tonto de cpu y un posible agujero de seguridad, así que mejor pararlo y santas pascuas.

Cada vez que instalamos un nuevo servicio en nuestra distribución favorita se nos instalarán los scripts necesarios de arranque y parada necesarios, incluso ya se habrán configurado de forma óptima para que no tengas que tocar nada. Por ello, aún no es necesario que sepamos escribir un script para configurarlo nosotros mismos, cosa que además se nos escapa a los conocimientos vistos en el curso. Con lo que aquí veamos tendremos mucho juego.

Las dos grandes vertientes.

En cuanto hablamos de la administración de servicios en Unix, y por consiguiente de Linux, existen dos grandes familias o vertientes: System V y BSD, su forma de organizar los runleves (que veremos más adelante) y el arranque y parada de servicios son muy diferentes.

Linux bebe de ambas corrientes, y no sólo en cuanto a administración de servicios se refiere, por lo que unas distribuciones usarán el sistema de scripts de System V como son Debian, Red hat, Suse, Mandriva... y otras usarán el sistema de scripts BSD, no estoy seguro pero creo que Gentoo y Slackware usan este sistema. o por lo menos antes sí que lo usaban.

En System V tenemos un script de control por daemon y en BSD tenemos un gran fichero por runlevel que arranca el sistema y pone en marcha los servicios. Muchos opinan que el sistema de scripts de System V es mucho mejor para uso diario que el sistema BSD, entre los que me incluyo. Según mi experiencia, los basados en System V son la inmensa mayoría y no tengo ni una Gentoo ni una Slackware instalada, con lo que sólo vamos a ver System V.

Antes de nada, los runlevels.

Que aún os queda un poquito de tostón, aunque este es importante y nos mete en faena, así que prestad atención.

Los runlevels en UNIX son estados del sistema donde se determinan qué demonios están activos y qué demonios están parados, algo así como los perfiles de uso de nuestras máquinas.

Según el Linux Standar Base, LSB para los amigos, y la tradición manda (ya sabemos que las tradiciones mandan más que los estándares), los runlevels son los siguientes:

0. Parar y/o apagar el ordenador.
1. Modo monousuario.
2. Multiusuario sin red.
3. Multiusuario con red.
4. No se usa.
5. Multiusuario con red y entorno gráfico (El que todos solemos usar por defecto).
6. Reiniciar la máquina.

Pero como no vivimos en un mundo perfecto y cada distribución los modifica según le vieneen gana, esto puede no ceñirse a la realidad de nuestra distribución y es aquí donde debemos investigar.

Y es aquí donde debemos visualizar un fichero de configuración muy importante, '/etc/inittab':

```
matados2k@imperio:~$ cat /etc/inittab
[... Parte eliminada de la salida ...]
# The default runlevel.
id:2:initdefault:

[... Parte eliminada de la salida ...]
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
matados2k@imperio:~$
```

Sólo he dejado lo que nos interesa del fichero en cuestión, ya que puede ser muy largo y no es el momento aún de aprender a usarlo.

Para empezar mi distribución es una Debian Etch, en las vuestras es normal que cambie pero hay que buscar lo mismo siempre, sea cual sea. Lo primero que buscamos es '`id:2:initdefault:`' que nos va a indicar el runlevel por defecto, y esto nos lo dice 'initdefault' y el '2' en este caso es el runlevel por defecto.

Lo siguiente que busco es una explicación en los comentarios, todo lo que vaya detrás de '#' es un comentario, es la explicación de los runlevels, ya veis lo diferente que es en Debian de lo que se supone que es lo estándar. Pero vamos, que de esto no se libra casi ninguna como podéis observar en este enlace: http://en.wikipedia.org/wiki/Runlevel#Standard_runlevels .

No en todas las distribuciones va a venir en el '/etc/inittab/' una explicación en los comentarios, así que en esos casos tendréis que tirar de documentación o de google como todo hijo de vecino.

Con esto ya tenemos o deberíamos tener claro qué scripts y para qué ' perfil de uso' vamos a tocar, y lo más importante, no tocar donde no debemos y mandarlo todo al carajo. Y vosotros no querréis cargaros algo, ¿verdad?

Pero ... ¿Dónde se encuentra lo que vamos o debemos tocar?.

"Ande andará", como dirían Cruz y Raya, pues esto también puede cambiar de unas distribuciones a otras, pero no preocuparse es fácil de encontrar.

Vamos a tener un directorio con todos los scripts de servicios y un directorio por cada runlevel con enlaces simbólicos a este primero siguiendo unas reglas que veremos a su tiempo, ahora estamos buscando cuáles son.

Según System V, el primer directorio que buscamos 'init.d' , y se llama así en todas, debe de estar en '/etc/rc.d/init.d', pero según la LSB debe de estar en '/etc/init.d', y en mi caso Debian los tiene en este último sitio. De cualquier forma, siempre debéis buscar un directorio llamado 'init.d' dentro de '/etc' o algún subdirectorio suyo.

Los directorios con enlaces simbólicos por cada runlevel tienen la siguiente forma 'rcX.d' donde 'X' es el número de runlevel que estamos buscando. En Debian están en '/etc/' directamente ('/etc/rc0.d','/etc/rc1.d'....), pero también los podéis encontrar en otras distribuciones bajo '/etc/rc.d/'.

Así que ya sabéis, buscad y localizad que no es nada difícil.

Preparados, listos, ¡YA!

Vamos a empezar a usar los scripts de 'init.d', y si sois como yo que tengo tanto KDE como GNOME en mi ordenador podréis seguir al pie de la letra el ejemplo, aunque este es válido para cualquier otro script.

Cada uno de estos escritorios usa gestor de sesión propio, KDE usa 'kdm' y GNOME usa 'dgm', y están tal cual con esos nombres dentro de 'init.d'. En mi caso, Debian sólo me instalaba GNOME con su 'gdm' y luego yo instalé KDE. Como 'gdm' no me gusta instalé además 'kdm' que es mi preferido. Pero 'gdm' seguía arrancado y lo paré. Vamos a ver cómo sería:

```
matados2k@imperio:~$ cd /etc/init.d
matados2k@imperio:/etc/init.d$ ./gdm
```

```
Usage: /etc/init.d/gdm {start|stop|restart|reload|force-reload}
matados2k@imperio:/etc/init.d$
```

Lo primero es que no nos vale con ejecutarlo y punto, tenemos una serie de parámetros, esto es, órdenes que darle, y suelen ser 5:

1. start. Para comenzar el servicio.
2. stop. Para parar el servicio.
3. restart. Para reinicializar el servicio.
4. reload. Para que el servicio cargue de nuevo la configuración.
5. force-reload. Para obligar al servicio que cargue la configuración de nuevo.

Los dos primeros siempre van a estar en cualquier script, el tercero es rarísimo que no esté, y los dos últimos puede que estén o no, e incluso puede darse la posibilidad de que algún script implemente algún parámetro más.

1 y 2 no necesitan explicación ninguna, 3 internamente suele estar implementado como primero haz 2 y luego haz 1, pero en algunos casos hay que hacer otras cosas de por medio para reiniciar el servicio sin problema.

4 lo que normalmente hace es decirle al servicio que lea de nuevo los ficheros de configuración y los aplique, esto no es siempre posible sin pararlo y en ese caso se utilizaría 5, al obligarlo puede que se produzcan rechazos de peticiones o que peticiones no finalicen correctamente durante el proceso.

A que ya lo vas pillando, pues te vas a quedar de momento con las ganas de seguir porque se acaba el espacio de esta entrega. Si no estás seguro de lo que haces o vas a hacer, espera a la entrega de la próxima semana.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 34. Administración de servicios (y II).

```
$ mkdir matter; cat >matter
matter: cannot create
```

Continuamos.

Sí, lo sé. Llego con un poco de retraso, pero es que una pandilla de turrones, polvorones y fiesta aderezada con algo de vacaciones, no me dejaban. Además, seguro que nadie me ha echado de menos.

Nos quedamos en la parada del servicio 'gdm' y lo realizaríamos así:

```
imperio:/etc/init.d# ./gdm stop
Stopping GNOME Display Manager: gdm.
imperio:/etc/init.d#
```

Y mucho cuidadín con no poner './' y no hace falta decir por qué a estar alturas, ¿verdad?. Ahora lo arranco:

```
imperio:/etc/init.d# ./gdm start
Not starting GNOME Display Manager; it is not the default display manager.
imperio:/etc/init.d#
```

Pero qué mala persona, ahora se niega... bueno, por si acaso tenéis bailes de ventanas gráficas por culpa de mi ejemplo, vamos con algo más dócil :

```
imperio:/etc/init.d# ./apache2 stop
Stopping web server (apache2)....
imperio:/etc/init.d# ./apache2 start
Starting web server (apache2)....
imperio:/etc/init.d#
```

Si es que no hay nada como hacer el indio para que los ejemplos salgan bien, ahora vamos a ver el reinicio, el recargar la configuración y el forzar dicha recarga:

```
imperio:/etc/init.d# ./apache2 restart
Forcing reload of web server (apache2)....
imperio:/etc/init.d# ./apache2 reload
Reloading web server config...11782
.
imperio:/etc/init.d# ./apache2 force-reload
Forcing reload of web server (apache2)....
imperio:/etc/init.d#
```

Por si alguien no sabe a estas alturas qué es apache, ya os digo que es el mejor servidor web que existe.

Definiendo el arranque y parada.

Y ya estamos en lo último para terminar la serie. Ya comentábamos que para cada runlevel teníamos un directorio con enlaces simbólicos. Éstos le indican a cada runlevel qué servicios tiene que arrancar y qué servicios tiene que parar.

Como en mi Debian el runlevel por defecto es el 2, voy a ver cuáles tengo:

```
imperio:/etc/init.d# cd ../rc2.d
imperio:/etc/rc2.d# ls
K99gdm              S20courier-authdaemon  S20openbsd-inetd  S89atd
README              S20courier-mta         S20samba          S89cron
S10sysklogd         S20cupsys              S20ssh            S90binfmt-support
S11klogd            S20dbus-1              S20webmin         S91apache2
S18portmap          S20firestarter         S21fam            S99kdm
S19autofs           S20inetd               S21mysql-ndb      S99rc.local
S19mysql-ndb-mgm    S20lisa                S21nfs-common     S99rmnologin
S19postgresql-8.1   S20lpd                 S25bluetooth      S99stop-bootlogd
S20acpid            S20makedev             S25bluez-utils
S20apmd             S20mysql               S89anacron
imperio:/etc/rc2.d#
```

Si sois observadores, el formato es que todos empiezan o bien por 'K' o bien por 'S', y no, README no es ningún script de servicio de lectura. Todo fichero que empiece por 'K' al arrancar la máquina en el runlevel determinado o al cambiar de un runlevel a otro, que en breve veremos cómo se hace, es parado.

Y claro, los más listos dirán “vale, pues con no ponerlo ya no se arranca”, pero la respuesta es fácil. ¿Y si venimos de un runlevel que sí cargaba el servicio?

Con eso ya podéis deducir para qué sirve empezar por 'S', efectivamente, es para decirle que tiene que arrancar un determinado servicio. Pero hay más, hay dos números a continuación, pues tienen la sencilla misión de indicar el orden de arranque o parada.

Fácil, ¿verdad?

Recapitulemos:

- Todos los script de administración de servicios se encuentran en 'init.d'.
- Esos script los podemos usar manualmente según nuestras necesidades.
- El arranque y parada de servicios de cada runlevel esta definido en un directorio llamado 'rcX.d' donde X es el número del runlevel que deseamos manipular.
- Esos directorios contienen únicamente enlaces simbólicos a los scripts contenidos en 'init.d'.
- Los enlaces tienen una forma especifica de nombrarlos ,'K' para parada, 'S' para arranque, un número para establecer el orden y el nombre del script que normalmente se corresponderá con

el nombre.

Ya vimos cómo crear enlaces simbólicos muchas entregas atrás, así que si no sabes seguir tu sólo en este momento deberías preguntarte si realmenteestas aprovechando bien el curso.

De runlevel a runlevel, chico esto se mueve.

Si necesitamos cambiar de un runlevel a otro, por ejemplo, por un runlevel que no tengaentorno gráfico y esto lo saben muy bien quienes han instalado alguna ver un driver de nvidia, no necesitaremos reiniciar cambiando qué runlevel arrancará por defecto. Podemos cambiar en cualquier momento con este sencillo comando:

```
init RUNLEVEL
```

Del 1 al 5 normalmente, y según nuestra configuración, cambiaremos entre los runlevel normales, si lo hacemos en 0 el ordenador se apagará y si lo hacemos en 6 se reiniciará y por lo menos en mi Debian si pongo una 's' entraría en monousuario.

Despedida.

Sí, lo sé, esta entrega ha sido corta, sólo espero que os resulte útil. De momento no sé con qué empezaré la próxima entrega, ya que al escribir estas líneas estoy muerto de cansancio y no me apetece pensarlo, así que... ¡Ah, sorpresa! Hasta la semana que viene.

matados2k@imperio:~$ init 0
...
...

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 35. Personalización del entorno (I).

```
% why not?
No match.
```

Introducción.

Antes de empezar con lo que es el shellscript o scripts de intérprete de comandos en español, nos conviene tratar el tema de algunos ficheros de arranque, variables de entorno y cambio de prompt.

Una parte de la filosofia de Unix es no intentar predecir las futuras necesidades de los usuarios, y para ello se intenta hacer fácil la configuración del entorno acorde a las necesidades de cada uno. Esto se consigue entre otras cosas con los ficheros de configuración. Gran parte de ellos son ficheros de inicio y muchos empiezan por '.' para hacerlos ficheros ocultos. En esta entrega vamos a ver algunos de ellos.

Tipos de shell.

El intérprete de comandos más usado es el bash sin duda, y nos centraremos mucho en él. Este intérprete de comandos puede funcionar dedistintas formas:

1. Intérprete de comando de ingreso. Es el modo en que arranca cuando ingresa por primera vez (hablamos sin entorno gráfico), debería ser el primer intérprete que se vea.
2. Intérprete de comando interactivo. Este es un intérprete de comando que nos presenta un prompt y espera la entrada de datos. Los intérpretes de comando de ingreso también son intérpretes de comando interactivos. Se puede tener un intérprete de comandos interactivo sin ingresar al sistema, por ejemplo ejecutando cualquier terminal en nuestro escritorio gráfico favorito (konsole, xterm, Gnome terminal ...)
3. Intérprete de comandos no interactivo. Se usan para ejecutar archivos de comandos, shellscripts, muy parecidos a los ficheros de procesamiento por lotes .BAT de ms-dos. Estos archivos de comandos funcionan como pequeños programas, mucho más lentos que los programas compilados pero también mucho más fáciles de escribir.

En el caso de bash, dependiendo del tipo de intérprete se ejecutarán distintos archivos para ponerlo en marcha:

Ingreso	Ejecución de .bash_profile
Interactivo	Ejecución de .bashrc
No interactivo	Ejecución del script indicado

Más ficheros de arranque y del entorno interesantes.

Hay más ficheros que nos van a interesar para el manejo de nuestro entorno de trabajo. De entre ellos vamos a listar algunos.

~/.bash_history	Historial de órdenes ejecutadas porel usuario.

/etc/X11/xinit	Script arrancado cuando una sesión gráfica arranca mediante xinit, 'startx'
/etc/X11/Xsession	Script usado cuando se arranca una sesión gráfica con algún gestor como 'kdm' o 'gdm'
/etc/profile	Arrancado cuando bash se usa como intérprete de comandos de ingreso para todos los usuarios
/etc/csh.login	Arrancado cuando tcsh se usa como intérprete de comandos de ingreso para todos los usuarios
~/.login	Arrancado cuando tcsh se usa como intérprete de comandos de ingreso
~/.tcshrc	Arrancado cuando tcsh se usa como intérprete de comandos interactivo
~/chrc	Es usado si .tcshrc no se encuentra
~/.kde/Autostart/	Todo fichero que esté dentro de este directorio lo intentará ejecutar KDE en el arranque del usuario.

¿Qué pasa si usamos sh?

```
matados2k@imperio:~$ ls -l /bin/sh
lrwxrwxrwx 1 root root 4 2006-12-24 10:43 /bin/sh -> bash
matados2k@imperio:~$
```

Como 'bash' tiene todo lo que 'sh', nos engañan con un enlace a 'bash'. Es más, 'bash' tiene mucha de las características útiles de 'ksh' y 'csh'.

Los alias.

Hace mucho, mucho tiempo en una entrega muy lejana, concretamente la tercera, ya hablábamos del comando alias, incluso aparecía algo tal que así:

> [...] *Nota: los alias que definamos se esfumarán cuando cerremos nuestra sesión o reiniciemos la máquina, pero no tengáis miedo. Ya explicaremos más adelante cómo definir las cosas permanentemente.* [...]

Pues bien, ese momento va a llegar en esta serie de entregas, y como se supone que ya sabéis para qué sirve alias no necesita explicación. Es más ya sabéis donde definir los alias permanentemente, en '/etc/profile' para todo el mundo y en '~/.bashrc' si es en concreto para algún usuario.

Las variables de entorno.

Otra cosa muy interesante a usar en '.bashrc' es la definición de variables de entorno. Ahora explicaremos esto de las variables de entorno.

Cada programa se ejecuta en un entorno, y este entorno está definido por el intérprete de comandos que lo llamó. El entorno está definido dentro de cada intérprete de comandos. Es de gran utilidad tener herramientas para manejar este entorno, que no son más que variables definidas dentro del intérprete de comandos.

Quiero ver mi entorno.

Hay un comando muy útil para esta tarea, y es:

env [opciones] [programa]

Este comando tiene la capacidad de ejecutar un programa modificando su entorno. Pero lo que nos interesa de primeras es que 'env' a secas nos dice qué entorno tenemos:

```
matados2k@imperio:~$ env
...
TERM=xterm
SHELL=/bin/bash
...
DESKTOP_SESSION=kde
PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
PWD=/home/matados2k
LANG=es_ES@euro
...
HOME=/home/matados2k
LANGUAGE=es_ES:es:en_GB:en
...
matados2k@imperio:~$
```

Seguro que a casi todos os sale una lista enorme, y además yo solo he dejado las que me interesa que veáis, porque algunas de ellas las vamos a ver en profundidad un poco más adelante.

Como opciones de 'env' vamos a ver dos. Para la primera, imaginaos que queremos ejecutar un comando en un entorno totalmente limpio, nuestra opción es '-i':

```
matados2k@imperio:~$ env -i pwd
/home/matados2k
matados2k@imperio:~$
```

Hemos ejecutado 'pwd' en un entorno totalmente limpio, cosa que a este comando en concreto le da lo mismo, pero probad a ejecutar por ejemplo 'env -i man man'. Seguro que si teníais las páginas del manual en español, todo habrá salido en inglés.

Como segunda opción vamos a ver '-u nombre', donde nombre es el nombre de la variable de entorno que queramos. Esto eliminará dicha variable para el entorno en esa ejecución:

```
matados2k@imperio:~$ env -u LANG pwd
/home/matados2k
matados2k@imperio:~$
```

Probad lo mismo con 'env -u LANG man man' ;)

Pero yo sólo quiero consultar una en concreto.

Imaginaos que no queréis ver todo vuestro entorno, que sólo os interesa saber una variable en concreto, un primer intento podría ser algo tal que así:

```
matados2k@imperio:~$ echo LANG
LANG
matados2k@imperio:~$
```

Ummm... necesito algo más, con eso sólo consigo que imprima lo que pongo. Necesito preceder el nombre de la variable con el símbolo '$' y ya lo tendremos.

```
matados2k@imperio:~$ echo $LANG
es_ES@euro
matados2k@imperio:~$
```

Ya está, ya sabemos cómo consultar el valor de una variable de entorno, cosa que nos puede dar mucha información sobre cómo tenemos configurado nuestro sistema.

Despedida.

El espacio de esta semana se acaba, y debéis esperar un poco a la siguiente para continuar con un tema que a la larga os puede interesar mucho, sobre todo cuando empecemos con shellcript. Hasta la semana que viene.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 36. Personalización del entorno (II).

```
% gotta light?
No match.
```

Continuamos, Yo quiero definir mis propias variables.

Para definir nuestras propias variables, que se usen en el entorno de ejecución, lo tenemos muy fácil. Es tan sencillo como:

Nombre=Valor

Donde nombre es el nombre de la variable y valor una cadena. Si necesitamos tener espacios en el valor debemos usar las comillas, ya sean dobles o simples.

```
matados2k@imperio:~$ FGH=9
matados2k@imperio:~$ echo FGH
FGH
matados2k@imperio:~$ echo $FGH
9
matados2k@imperio:~$
```

Hay que acordarse siempre de usar el símbolo '$' para ver el contenido de la variable, para cambiar el valor de la variable simplemente la volvemos a definir:

```
matados2k@imperio:~$ FGH="Soy Matados2k"
matados2k@imperio:~$ echo ¿Quién eres tu? $FGH
¿Quién eres tu? Soy Matados2k
matados2k@imperio:~$
```

Ahora podéis observar cómo he cambiado su valor y he añadido comillas para poder darle el valor de cadenas con espacios. Es muy importante no dejar espacios entre el nombre de la variable, el '=' y el valor, o si no:

```
matados2k@imperio:~$ FGH = 9
bash: FGH: command not found
matados2k@imperio:~$ FGH= 9
bash: 9: command not found
matados2k@imperio:~$ FGH=Hola soy yo
bash: soy: command not found
matados2k@imperio:~$
```

Pero definir así una variable tiene la limitación de que sólo funcionará en esa variable en esa ejecución de bash, y si lanzamos un proceso que a su vez lance procesos hijos no tendrán acceso a esa

variable:

```
matados2k@imperio:~$ FGH=9
matados2k@imperio:~$ bash
matados2k@imperio:~$ echo $FGH

matados2k@imperio:~$ exit
exit
matados2k@imperio:~$ echo $FGH
9
matados2k@imperio:~$
```

En este ejemplo lo vemos claramente, primero definimos la variable, arrancamos otra ejecución de 'bash' dentro de 'bash' e intentamos visualizar la variable, la nueva ejecución de 'bash'no ha heredado la variable... si salimos de ella y volvemos a la ejecución de 'bash' original sí podemos visualizarla.

Voy a llegar a todos lados, lo vais a ver.

Pues así esto no es de mucha ayuda, así que vamos a ver una nueva forma de definir nuestra variable o simplemente de ampliarla.

export variable[=valor]
declare -x variable[=valor]

En este caso es opcional definirle un valor porque la variable puede haberse declarado con anterioridad, tanto la primera como la segunda forma hacen lo mismo y la más usada es la primera:

```
matados2k@imperio:~$ export FGH
matados2k@imperio:~$ bash
matados2k@imperio:~$ echo $FGH
9
matados2k@imperio:~$
```

Veamos lo mismo pero con declare:

```
matados2k@imperio:~$ export FGH=9
matados2k@imperio:~$ declare -x HGF=6
matados2k@imperio:~$ echo $HGF
6
matados2k@imperio:~$ bash
matados2k@imperio:~$ echo $HGF
6
matados2k@imperio:~$
```

Pues ahora quiero eliminarlas.

También podemos destruir las variables de lasiguiente forma:

unset nombre

Veamos un ejemplo:

```
matados2k@imperio:~$ unset HGF
matados2k@imperio:~$ unset FGH
matados2k@imperio:~$ echo $FGH $HGF


matados2k@imperio:~$
```

En definitiva, rápido, fácil y sencillo.

Un momento, para el carro que te he calado.

Si hacéis pruebas y porejemplo ejecutáis un terminal exportáis una variable y luego lo cerráis, se pierde la variable o bien abrís dos sesiones de consola en distintos terminales, veréis que lo que se exporta en una no se ve en la otra. ¿Esto por qué?

Porque la exportación se produce de padres a hijos. Entonces, ¿cómo definimos una variable para todos y para siempre? Pues en los ficheros de arranque como '~/.bashrc' para todas las sesiones de un usuario y en '/etc/profile' si quieres que sea efectivo para todo el mundo y todo solucionado.

Muy bien, y todo este rollo ¿Para qué me vale?.

Pues por ejemplo para cambiar una variable de entorno muy importante, para personalizar nuestro propio entorno, valga la redundancia. Esta variable es una de la más importantes y se llama PATH:

```
matados2k@imperio:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
matados2k@imperio:~$
```

PATH es un listado de rutas separadas por ':' donde los intérpretes de comando buscarán los comandos a ejecutar. Por ejemplo, cuando ejecutamos 'ls' , lo primero que hará 'bash' es buscarlo en '/usr/local/bin', como no lo encuentra lo buscará '/usr/bin' donde tampoco está, y finalmente lo encontrará en '/bin' donde sí está y lo ejecuta. ¿Qué pasaría si elimino '/bin' de la ruta y ejecuto 'ls'?

```
matados2k@imperio:~$ PATH=/usr/local/bin:/usr/bin:/usr/bin/X11:/usr/games
matados2k@imperio:~$ ls
bash: ls: command not found
matados2k@imperio:~$
```

Vaya, ahora tendría que ejecutar '/bin/ls' para que fuera efectivo:

```
matados2k@imperio:~$ /bin/ls
```

```
2805-fantasy.jpg
...
XF86Config-4ACTUAL
xmms-playlist
matados2k@imperio:~$ PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
matados2k@imperio:~$
```

Así que mejor lo dejo como estaba :). Si queremos que mire en algún directorio más aparte de los ya definidos, la forma correcta sería:

export PATH=$PATH:Nueva_ruta

Vamos a ver un ejemplo donde me defino un nuevo directorio con un script muy simple:

```
matados2k@imperio:~$ cd curso
matados2k@imperio:~/curso$ mkdir miscomandos
matados2k@imperio:~/curso$ echo echo Quien quiere las come quien no las deja >
./miscomandos/lentejas
matados2k@imperio:~/curso$ chmod +x ./miscomandos/lentejas
matados2k@imperio:~/curso$ ./miscomandos/lentejas
Quien quiere las come quien no las deja
matados2k@imperio:~/curso$ PATH=$PATH:/home/matados2k/curso/miscomandos/
matados2k@imperio:~/curso$ lentejas
Quien quiere las como quien no las deja
matados2k@imperio:~/curso$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:/home/matados2k/curso/misco
mandos/
matados2k@imperio:~/curso$
```

Como veis un ejemplo muy estúpido que nos muestra de forma muy clara y contundente la gran utilidad que tiene manejar estas variables de entorno y saber dónde definirlas para que queden para siempre.

Despedida.

¡Ohh!... Se nos vuelve a acabar el espacio de esta semana justo cuando viene lo mejor, pero qué malo que soy. En la siguiente entrega vamos a profundizar más en esta variable y otras muy interesantes que nos permitirán modificar la información de nuestro 'prompt', no me faltéis.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 37. Personalización del entorno (y III).

> % ^How did the sex change^ operation go?
> Modifier failed.

Seguimos con el PATH.

Otra utilidad que se puede hacer con la variable PATH es saltarnos './' para la ejecución de un ejecutable que no esté en la ruta. Para muestra:

```
matados2k@imperio:~$ cd curso/miscomandos/
matados2k@imperio:~/curso/miscomandos$ echo echo Hola Caracola > hola
matados2k@imperio:~/curso/miscomandos$ chmod +x hola
matados2k@imperio:~/curso/miscomandos$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
matados2k@imperio:~/curso/miscomandos$ hola
bash: hola: command not found
matados2k@imperio:~/curso/miscomandos$ PATH=$PATH:./
matados2k@imperio:~/curso/miscomandos$ hola
Hola Caracola
matados2k@imperio:~/curso/miscomandos$
```

Ahora hemos conseguido el comportamiento típico de ms-dos. De hecho, hay distribuciones que ya traen preparado esto. Pero personalmente, esto es algo que puede producir un agujero en la seguridad, en este caso puede que no, pero ahora veamos otro ejemplo:

```
matados2k@imperio:~/curso/miscomandos$ echo echo Eres marica y el culo te pica >
ls
matados2k@imperio:~/curso/miscomandos$ chmod +x ls
matados2k@imperio:~/curso/miscomandos$ ls
hola  lentejas  ls
matados2k@imperio:~/curso/miscomandos$ PATH=./:$PATH
matados2k@imperio:~/curso/miscomandos$ ls
Eres marica y el culo te pica
matados2k@imperio:~/curso/miscomandos$
```

Aquí podemos observar cómo poniendo './' justo como el primero en la ruta podemos suplantar uno de los ejecutables más usados en GNU/Linux y en cualquier Unix. En este caso sólo nos han colado un mensaje muy molesto y bastante idiota.

Ahora supongamos que tenemos un scritpt que borre el disco duro desde el raíz y de forma silenciosa, probablemente si estamos desde un usuario sin permiso de root se produzca entre nada y poco daño al sistema, dado que necesita permisos.

Pero ahora supongamos que mejoras el script y lo primero que haces es comprobar si tiene permisos de root, si no es así, ejecuta 'ls' con los parámetros indicados pero si lo es hace el borrado desde '/' de forma silenciosa. Seguimos suponiendo, y supongamos que yo tengo un amigo llamado Mandito y éste tiene un ftp en el que yo puedo subir cosas, y Manolito dice que Ricky Martin es mejor que Los Mojinos Escozíos, como eso es una aberración me cabreo y le coloco el script en su ftp.

Resulta que Manolito entra en su máquina donde tiene el servidor de ftp y entra como root porque tiene que hacer tareas de mantenimiento, entra en mi directorio y ejecuta 'ls' para ver qué le he subido. Como Manolito es muy cómodo pero algo malo como administrador tenía el PATH como en el segundo ejemplo, y como resultado se ha comido un exploit que le ha borrado el disco duro de la manera más tonta posible, que es no reconocer que Los Mojinos son 'el mejón grupo' del mundo mundial.

Personalmente, y fuera de bromas, yo soy de los que prefieren './' fuera de cualquier parte de la variable PATH.

La variable PS1

Como es evidente, no voy a hablar de la Play Station, sino de una de las variables que nos van a permitir personalizar nuestro prompt para dejarla con la información que a nosotros nos guste.

Para ver la configuración que tenemos, la visualizamos:

```
matados2k@imperio:~$ echo $PS1
\u@\h:\w\$
matados2k@imperio:~$
```

Y ahora toca traducir qué es lo que hace eso, y para ello usamos la siguiente información:

- \a carácter de campana ASCII (07)
- \d la fecha en formato día mes día (p.ej., mar may 26)
- \e caracter de escape ASCII (033)
- \h el nombre del host hasta el primer «.»
- \H el nombre del la máquina completo (FQDN)
- \n caracter de nueva línea
- \r retorno de carro
- \s el nombre del shell, el nombre base de $0 (el fragmento que sigue a la última barra)
- \t la hora actual en formato 24-horas HH:MM:SS
- \T la hora actual en formato 12-horas HH:MM:SS
- \@ la hora actual en formato 12-horas AM/PM
- \u el nombre de usuario del usuario actual
- \v la versión de bash (p.ej., 2.0)
- \V la versión del paquete del bash, versión + patch-level (p.ej., 2.00.0)
- \w el directorio actual de trabajo
- \W el nombre base del directorio actual de trabajo
- \! el número del comando actual en el histórico
- # el número de comando del comando actual
- \$ si el UID efectivo es 0, un #; en otro caso, $
- \nnn el caracter correspondiente al número en octal nnn
- \\ una contrabarra
- \[inicio de una secuencia de caracteres no imprimibles que pueden usarse para incrustar una secuencia de control del terminal en el prompt.

```
- \] fin de una secuencia de caracteres no imprimibles
```

Lo primero es saber que todos los símbolos con significado especial llevan el carácter de escape '\'. Mirando la tabla traducimos que primero debe poner el nombre del usuario activo, seguido del símbolo @, el nombre del host que es el nombre de mi máquina, seguido de :, el nombre del directorio donde estoy y el símbolo final.

Ahora voy a crearme uno nuevo:

```
matados2k@imperio:~$ AUXILIAR=$PS1
matados2k@imperio:~$ PS1="El mejor es \u y son las \t\a :P "
El mejor es matados2k y son las 19:51:01 :P PS1=$AUXILIAR
matados2k@imperio:~$
```

Como veis podéis hacer muchas combinaciones y ponerlo avuestro gusto.

La guinda del pastel

Por último vamos a ver un fichero más, ya que personalizando podríamos hacer muchas más entregas, así que decido poner fin con esto a esta serie de entregas.

Si os logueáis desde una consola sin ser bajo entorno gráfico veréismuchas veces mensajes de bienvenida parecidos a este:

```
Linux imperio 2.6.17-2-686 #1 SMP Wed Sep 13 16:34:10 UTC 2006 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
matados2k@imperio:~$
```

El fichero que contiene lo que va a mostrase como bienvenida es 'etc/motd', así que puedes poner lo que quieras y ver los cambios:

```
matados2k@imperio:~$ su
Password:
imperio:/home/matados2k# cd /etc
imperio:/etc# cp motd motd_copia
imperio:/etc# echo Vendo Ford Fiesta 1.1 > motd
imperio:/etc# echo En buen estado y de color rosa >> motd
imperio:/etc# echo Con resonador y ruedas de 205/35 15\" y mega alerón >> motd
imperio:/etc# echo Horteras al poder >> motd
imperio:/etc#
```

Y ojo, que esto lo verá cualquier usuario al entrar en la máquina, no solo tú, así que a todos les llegará mi bonito mensaje y de paso a ver si cuela:

```
Vendo Ford Fiesta 1.1
En buen estado y de color rosa
Con resonador y ruedas de 205/35 15" y mega aleron
matados2k@imperio:~$
```

Espero que si probáis a poner tonterías luego restauréis los ficheros como yo hago:

```
imperio:/etc# cp motd_copia motd
imperio:/etc# cat motd
Linux imperio 2.6.17-2-686 #1 SMP Wed Sep 13 16:34:10 UTC 2006 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
imperio:/etc#
```

Despedida.

Pues sin más me despido hasta la próxima semana, donde ya veremos si empiezo a explicar la programación de shell script. No me falléis.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 38. Programación de tareas (I).

> % ^How did the sex change operation go?
> Bad substitute.

Comenzamos.

Bien, ya estamos de nuevo, ya comenté que comenzaría con Shell Script pero creo que la programación de tareas puede ser interesante verla antes, ya que le dará mucha potencia a los scripts que aprendamos a hacer en la próxima serie de entregas.

A lo largo de estas entregas vamos a aprender a usar varias herramientas para programar tareas, aunque en el fondo sean sólo dos, y de ellas luego dos interfaces gráficas, y como siempre una para KDE y otra para GNOME.

Sólo por una vez.

Se nos puede dar el caso de querer ejecutar una tarea una sola vez pero en un momento determinado. Para realizar esto usamos el siguiente comando:

at [opciones] TIME

Este comando ejecuta comandos a una determinada hora, donde time es el tiempo en el formato "HH:MM" donde HH es hora y MM minutos. Vamos a ejecutarlo:

```
matados2k@imperio:~/curso$ at
Garbled time
matados2k@imperio:~/curso$ at 11:00
warning: commands will be executed using /bin/sh
at>
```

Como vemos, es necesario indicar siempre el tiempo y se nos queda a la espera de que le introduzcamos los comandos. Bien, le metemos uno sencillo y para salir pulsamos 'ctrl+d':

```
at> ls > ls.txt
at> (aquí pulso Ctrl+d) <EOT>
job 5 at Thu Feb 15 11:00:00 2007
matados2k@imperio:~/curso$
```

Como no se nos muestra nada una vez se ejecuta el comando, una vez pasado el tiempo, vamos a comprobar el resultado:

```
matados2k@imperio:~/curso$ ls ls.txt
ls.txt
matados2k@imperio:~/curso$ cat ls.txt
```

```
aMule-2.0.0rc4a-1.i686-FC.rpm
aMule-2.0.0-rc5_fc1.i686.rpm
...
wxGTK-devel-2.4.2-1.i386.rpm
matados2k@imperio:~/curso$
```

Si lo que quieres es indicarle directamente un script y olvidarte de pulsar 'ctrl+d' puedes usar la opción '-f ' y listarle el comando que necesites:

```
matados2k@imperio:~/curso$ cd miscomandos/
matados2k@imperio:~/curso/miscomandos$ ls
hola  lentejas  ls
matados2k@imperio:~/curso/miscomandos$ echo ls -l \> lista.txt  > listar
matados2k@imperio:~/curso/miscomandos$ chmod +x listar
matados2k@imperio:~/curso/miscomandos$ cat listar
ls -l > lista.txt
matados2k@imperio:~/curso/miscomandos$ at -f listar 11:14
warning: commands will be executed using /bin/sh
job 8 at Thu Feb 15 11:14:00 2007
matados2k@imperio:~/curso/miscomandos$ cat lista.txt
total 16
-rwxr-xr-x 1 matados2k matados2k 19 2007-01-27 16:48 hola
-rwxr-xr-x 1 matados2k matados2k 45 2007-01-14 19:40 lentejas
-rwxr-xr-x 1 matados2k matados2k 18 2007-02-15 11:13 listar
-rw-r--r-- 1 matados2k matados2k  0 2007-02-15 11:14 lista.txt
-rwxr-xr-x 1 matados2k matados2k 35 2007-01-27 16:52 ls
matados2k@imperio:~/curso/miscomandos$
```

Expliquemos paso por paso: primero creo un script redireccionando la salida del comando echo a un fichero, donde uso '\' para escapar '>' y no lo interprete como redirección. Le doy permisos de ejecución, lo programo y compruebo su correcta ejecución.

¡Qué memoria la mía!.

Mira que ando mal de la memoria, programo las tareas con 'at' y luego se me olvida qué era lo que mandé y a qué hora se ejecutaba, necesito el comando:

atq

Este comando nos lo recuerda fácilmente:

```
matados2k@imperio:~/curso/miscomandos$ at -f listar 15:00
warning: commands will be executed using /bin/sh
job 9 at Thu Feb 15 15:00:00 2007
```

```
matados2k@imperio:~/curso/miscomandos$ at -f listar 18:00
warning: commands will be executed using /bin/sh
job 10 at Thu Feb 15 18:00:00 2007
matados2k@imperio:~/curso/miscomandos$ atq
9       Thu Feb 15 15:00:00 2007 a matados2k
10      Thu Feb 15 18:00:00 2007 a matados2k
matados2k@imperio:~/curso/miscomandos$
```

Como veis nos da todo lo que necesitamos , el número de la tarea (que nos va a ser útil), la fecha de la futura ejecución y el usuario.

Quiero desprogramar.

Un buen ejemplo para querer eliminar una tarea programada con 'at' puede ser que hayas sido tan gracioso de programar 'rm -r /' con permisos de root y claro, luego puede que no te siente bien cargarte toda tu información. Menos mal que existe:

atrm Números_de_tareas

El número de tarea nos lo da siempre el comando 'at' en las líneas como esta: '`job 10 at Thu Feb 15 18:00:00 2007`' o bien lo sacamos como hemos dicho antes con 'atq'.Como ya tengo los números de tarea, voy a eliminar las tareas pendientes de ejecutarse:

```
matados2k@imperio:~/curso/miscomandos$ atq
9       Thu Feb 15 15:00:00 2007 a matados2k
10      Thu Feb 15 18:00:00 2007 a matados2k
matados2k@imperio:~/curso/miscomandos$ atrm 9 10
matados2k@imperio:~/curso/miscomandos$ atq
matados2k@imperio:~/curso/miscomandos$
```

Soy un cotilla.

Como puede que tengamos una máquina con varios usuarios, puede que alguno de ellos use 'at' y quieras saber qué es lo que va a ejecutar, y para cotillear cuál es el entorno del usuario y qué ejecutará, vamos a usar la opción '-c' seguido del número de tarea y sin usar TIME para que nos lo chive:

```
matados2k@imperio:~/curso/miscomandos$ at -f listar 15:00
warning: commands will be executed using /bin/sh
job 11 at Thu Feb 15 15:00:00 2007
matados2k@imperio:~/curso/miscomandos$ atq
11      Thu Feb 15 15:00:00 2007 a matados2k
matados2k@imperio:~/curso/miscomandos$ at -c 11
#!/bin/sh
# atrun uid=1000 gid=1000
```

```
# mail matados2k 0
umask 22
KDE_MULTIHEAD=false; export KDE_MULTIHEAD
...
OLDPWD=/home/matados2k/curso; export OLDPWD
cd /home/matados2k/curso/miscomandos || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
ls -l > lista.txt

matados2k@imperio:~/curso/miscomandos$
```

Como puedes observar, nos da toda toda la información que podamos necesitar saber y más, para conocer qué se va a ejecutar y predecir con exactitud qué puede pasar.

Cuestión de permisos.

El superusuario (root) siempre puede utilizar estos comandos. Para otros usuarios, los permisos para utilizarlos están determinados en los ficheros '/etc/at.allow' y '/etc/at.deny'.

Si el fichero '/etc/at.allow' existe, sólo los usuarios cuyos nombres estén mencionados aquí tienen permiso para utilizar 'at'. Si '/etc/at.allow' no existe, se chequea '/etc/at.deny' y entonces todos los usuarios cuyos nombres no estén mencionados allí tienen permiso para utilizar 'at'. Si ninguno de los ficheros existe, sólo el superusuario puede utilizar 'at'. Un '/etc/at.deny' vacío significa que todo usuario puede utilizar estos comandos, esta es la configuración por defecto.

Veamos la configuración que tengo:

```
matados2k@imperio:~/curso/miscomandos$ cat /etc/at.allow
cat: /etc/at.allow: No existe el fichero o el directorio
matados2k@imperio:~/curso/miscomandos$ cat /etc/at.deny
cat: /etc/at.deny: Permiso denegado
matados2k@imperio:~/curso/miscomandos$ su
Password:
imperio:/home/matados2k/curso/miscomandos# cat /etc/at.deny
alias
backup
bin
...
sys
www-data
imperio:/home/matados2k/curso/miscomandos#
```

Despedida.

Vamos a cortar aquí para no dejar el siguiente comando a medias. Como veis, 'at' y los demás comandos vistos en esta entrega se nos pueden quedar cortos y necesitamos algo conmucha más potencia... Que es precisamente lo que veremos en la siguiente entrega, nos vemos pronto.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 39. Programación de tareas (II).

```
% drink bottle: cannot open
opener: not found
```

Continuamos.

Continuamos con lo visto en la entrega anterior, es muy probable que se nos quede corto y que necesitemos algo mucho más potente y versátil, para eso tenemos una herramienta muy útil llamada "cron", y en esta entrega nos vamos a dedicar a usarla.

¿Qué es cron?

Cron es un demonio para ejecutar tareas programadas, que será de gran ayuda tanto para administradores de sistemas como para usuarios comunes.

Normalmente, este demonio se encuentra ya instalado y arrancado, en caso contrario ya sabéis cómo actuar porque el arranque y parada e instalación ya lo hemos visto.

El uso más simple de cron

Para los que sean mas cómodos o que simplemente no necesiten más que algo que se ejecute cada hora, todos los días, todas las semanas o meses, están de suerte porque al menos en Debian existen cuatro directorios que cumplen con lo que necesitamos, y son:

- /etc/cron.hourly - Para cada hora
- /etc/cron.daily - Para cada día
- /etc/cron.weekly - Para cada Semana
- /etc/cron.monthly - Para cada Mes

Con esto sólo nos basta poner un ejecutable (o enlace a un ejecutable) o script para que se ejecute automáticamente, aunque este es el método más sencillo no permite en absoluto la personalización.

Si estáis en otros sistemas puede que no tengáis esto, pero es fácil de hacer, lo primero es crear los directorios, con los permisos del listado:

```
drwxr-xr-x  2 root   root     4096 2007-01-28 09:31 cron.daily
drwxr-xr-x  2 root   root     4096 2007-01-28 09:30 cron.hourly
drwxr-xr-x  2 root   root     4096 2007-01-28 09:30 cron.monthly
drwxr-xr-x  2 root   root     4096 2007-02-08 22:19 cron.weekly
```

Y lo siguiente, añadir estas líneas al fichero '/etc/crontab', que veremos más adelante para qué sirve:

```
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.weekly )
```

```
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --
report /etc/cron.monthly )
```

Sólo necesitaremos con lo anterior tener instalado 'anacron'.
Yo quiero más, mucho más

Aun así, el no poder personalizar aún más la programación de tareas puede ser un gran inconveniente, por lo que necesitamos seguir aprendiendo y utilizar 'cron' de la mejor forma posible.

Para ello, debemos definir las tareas dentro de archivos llamados 'crontab', y para ello hay dos formas: la genérica, donde sólo puede escribir el root quees el fichero antes nombrado '/etc/crontab', y la segunda son los ficheros 'crontab' por usuario dentro del directorio '/var/spool/cron/crontabs/' con el nombre del usuario, como por ejemplo '/var/spool/cron/crontabs/matados2k' y el grupo debe ser 'crontab':

```
imperio:/var/spool/cron/crontabs# ls -l
total 4
-rw------- 1 matados2k crontab 208 2007-02-20 13:21 matados2k
imperio:/var/spool/cron/crontabs#
```

¿Cómo rellenar estos archivos?

Estos archivos tienen dos cosas, variables y tareas. Las variables son cuatro y pueden o no aparecer, yo de hecho no las voy a usar, pero son estas:

- SHELL - Indica el intérprete de comandos a usar.
- PATH - Indica el PATH a usar que ya vimos en otras entregas .
- MAILTO - Indica el usuario al cual notificar las salidas generadas por las tareas.
- HOME - Indica el directorio personal del usuario

Un ejemplo de configuración de estas variables puede ser:

```
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
MAILTO=matados2k
HOME=/home/matados2k
```

Y las tareas, que es lo que tiene miga y lo que más nos interesa. El formato es el siguiente:

minuto hora día_del_mes mes día comando

Bastante auto explicativo, ¿verdad? Veamos qué valores valen para cada campo:

Campo	Valores permitidos
minuto	0-59
hora	0-23

día_del_mes	1-31
mes	1-12 (o el nombre en inglés)
día	0-7 (o el nombre en inglés, 0 ó 7 es Domingo)

Si os fijáis tiene mucho sentido que tanto 0 como 7 sea domingo, ya que para los anglosajones el primer día de la semana es domingo y usarán del 0 al 6, y nosotros usaremos del 1 al 7 y todo funcionará a las mil maravillas.

Para decir que se ejecute para todos sus posibles valores en cada campo de la tabla usaremos '*', para pasarle una lista de valores los separaremos por comas, por ejemplo lunes, miércoles y jueves sería '1,3,5'. Para un rango se utiliza un guión, por ejemplo de lunes a viernes '1-5', y por último periodos para los cuales usaremos '/', por ejemplo cada 15 minutos '*/15'.

Como ejemplo vamos a definir una tarea, que será para el comando 'ls -l /home/matados2k > /home/matados2k/lista.txt en los siguientes tiempos:

A las 8 de la mañana durante todos los días:

```
0 8 * * *   ls -l /home/matados2k > /home/matados2k/lista.txt
```

Cada 15 minutos los domingos:

```
*/15 * * * 7   ls -l /home/matados2k > /home/matados2k/lista.txt
```

Cada 2 horas los lunes y miércoles:

```
* */2 * * 1,3   ls -l /home/matados2k > /home/matados2k/lista.txt
```

Los días 1 de cada mes:

```
0 0 1 * *   ls -l /home/matados2k > /home/matados2k/lista.txt
```

De 12 de la noche a 8 de la mañana cada 30 minutos:

```
0/30 0-8 * * *   ls -l /home/matados2k > /home/matados2k/lista.txt
```

Si os paráis a pensar no es nada difícil crear un fichero válido para programar tareas, y las posibles combinaciones para conseguir una programación de tareas son geniales ¿Se os ocurre alguna combinación que no sea posible? (Y no vale empezar con cosas raras como ciclos lunares).

Y otra vez cuestión de permisos.

El superusuario (root) siempre puede utilizar estos comandos. Para otros usuarios, los permisos para utilizarlos están determinados en los ficheros '/etc/cron.allow' y '/etc/cron.deny'.

Si el fichero '/etc/cron.allow' existe, sólo los usuarios cuyos nombres estén mencionados aquí tienen permiso para utilizar 'cron'. Si '/etc/cron.allow' no existe, se chequea '/etc/cron.deny' y entonces todos los usuarios cuyos nombres no estén mencionados allí tienen permiso para utilizar 'cron'. Si ninguno de los ficheros existe, sólo el superusuario puede utilizar 'cron'. Un '/etc/cron.deny' vacío significa que todo usuario puede utilizar estos comandos, esta es la configuración por defecto (aunque esto puede cambiar en otros sistemas).

Como lo tengo yo configurado:

```
imperio:/home/matados2k# cat /etc/cron.allow
cat: /etc/cron.allow: No existe el fichero o el directorio
imperio:/home/matados2k# cat /etc/cron.deny
cat: /etc/cron.deny: No existe el fichero o el directorio
imperio:/home/matados2k#
```

Despedida.

Pues con esto terminamos por esta semana y os espero la siguiente con las alternativas gráficas para configurar la programación de tareas, que por debajo por supuesto usan 'cron'. Y mira por donde, será la entrega 40. Un buen número para quizás preparar un recopilatorio de entregas muy completito en pdf.

CURSO DESDE 0 DE GNU/LINUX. Versión 2.

Entrega 40. Programación de tareas (y III).

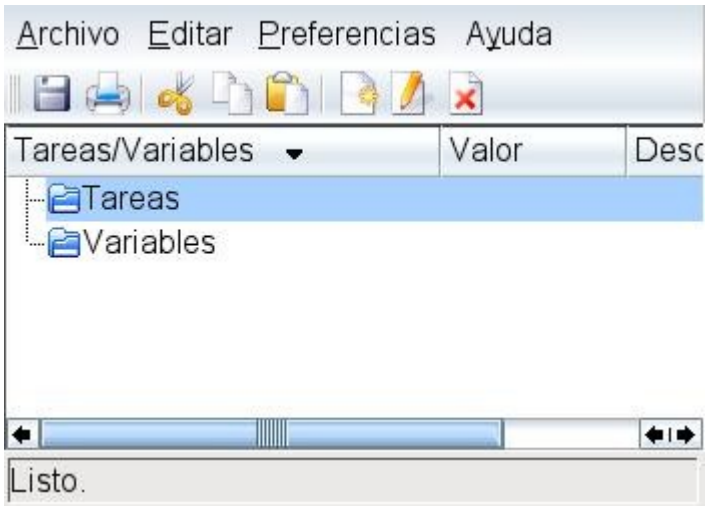
```
% look into "my eyes"
look: cannot open my eyes
```

Para finalizar.

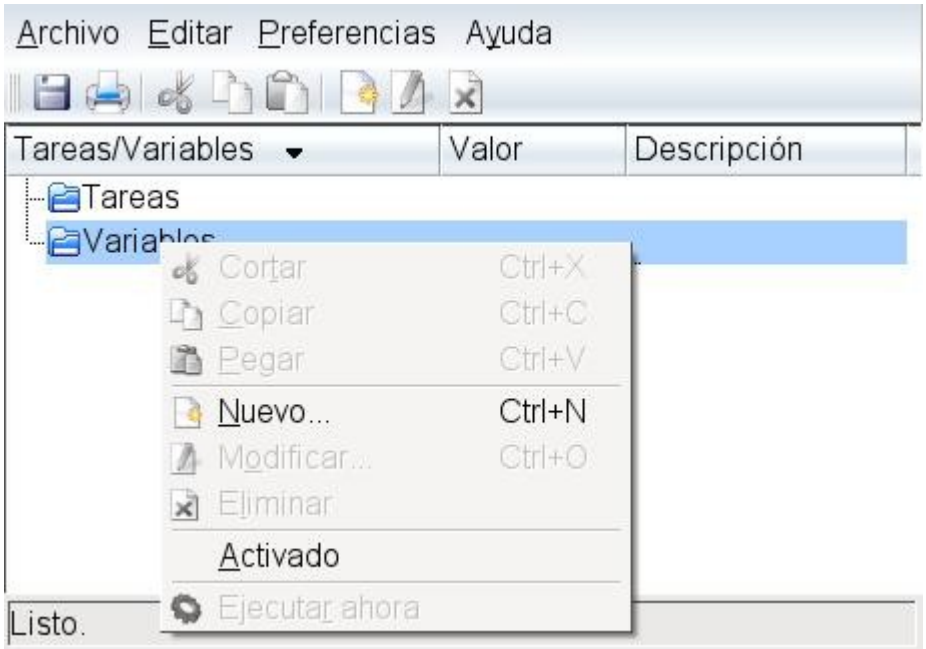
Para finalizar esta serie de entregas vamos a ver las herramientas gráficas que podemos usar, como siempre una para KDE y otra para Gnome.

Kcron.

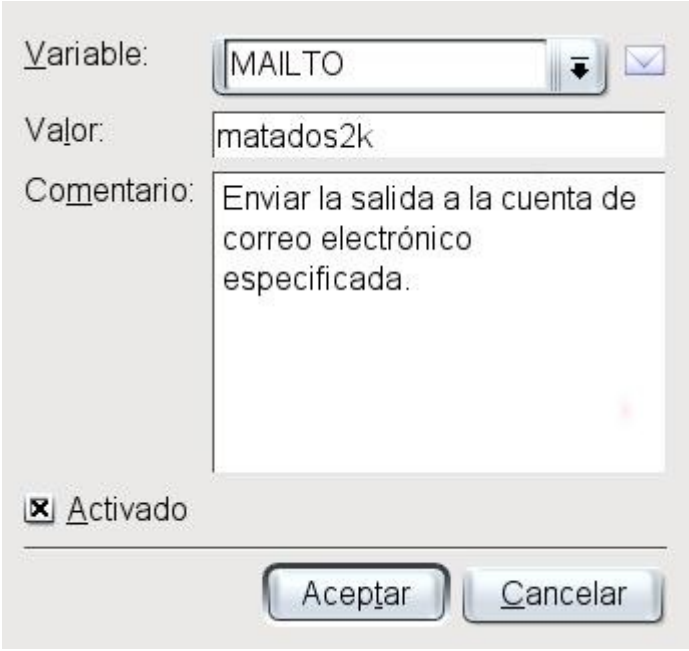
Esta es la alternativa para kde como su nombre sugiere, una vez abierto se nos muestra así:

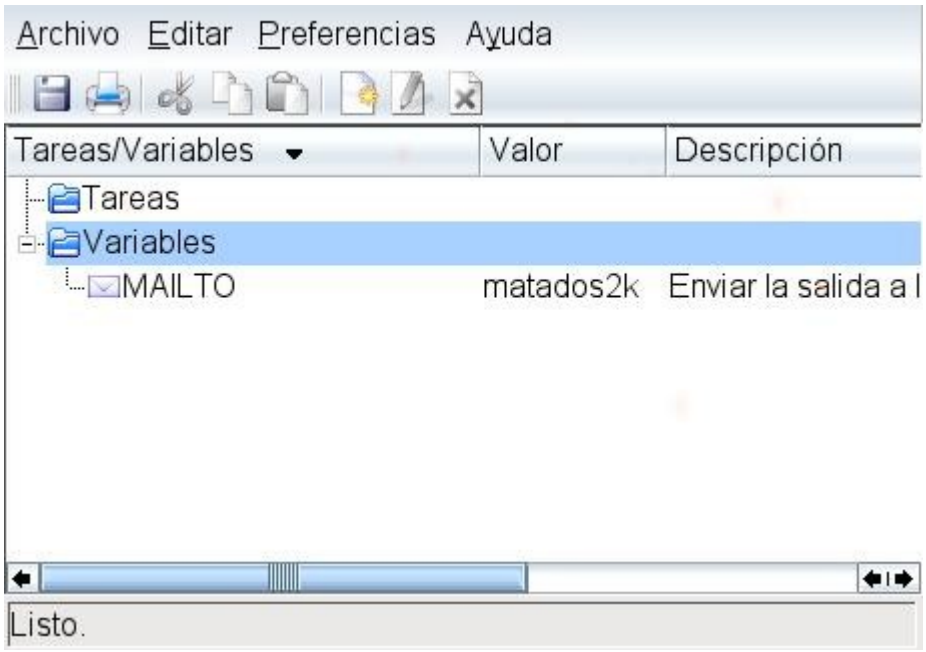


Vemos bien diferenciadas la parte referente a las variables y a las tareas. Para crear una variable con el segundo botón del ratón en la zona de variables elegimos Nuevo. También vale seleccionar la zona de variables y pulsar en el botón de nuevo.

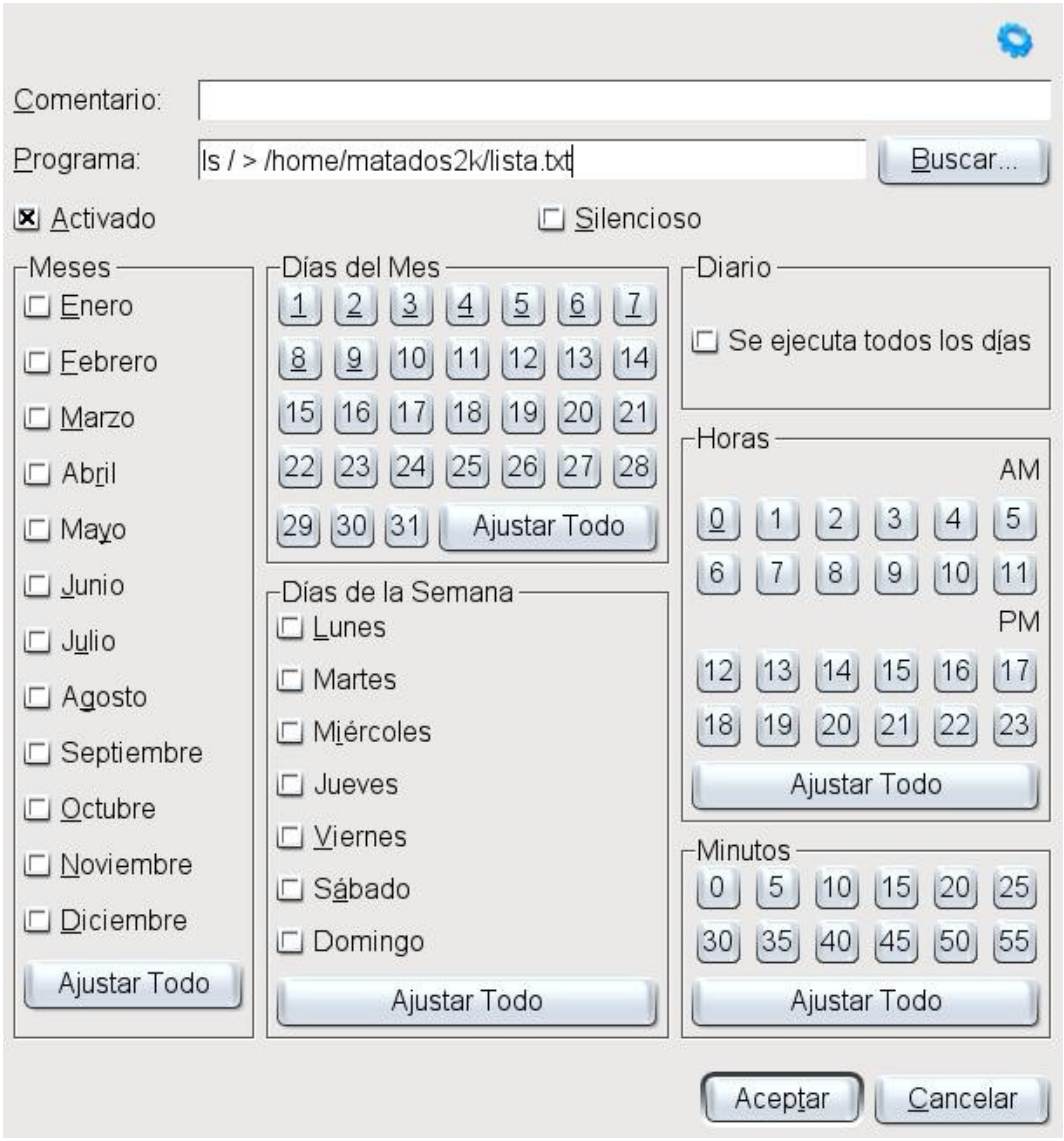


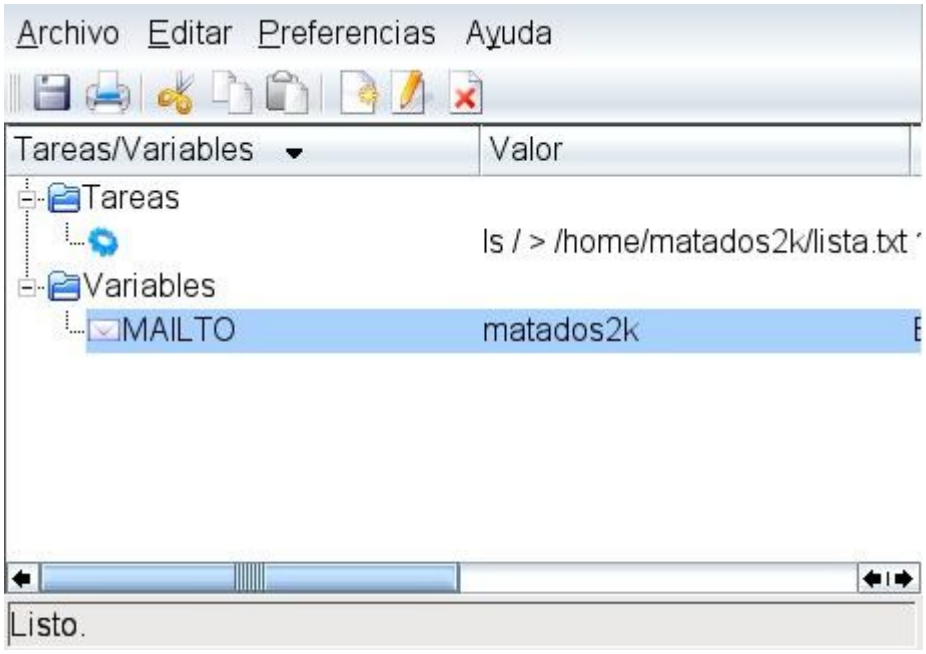
Simplemente rellenamos cualquiera de las variables y aceptamos:





Ahora realizamos el mismo paso para las tareas y se nos muestre una pantalla como esta:

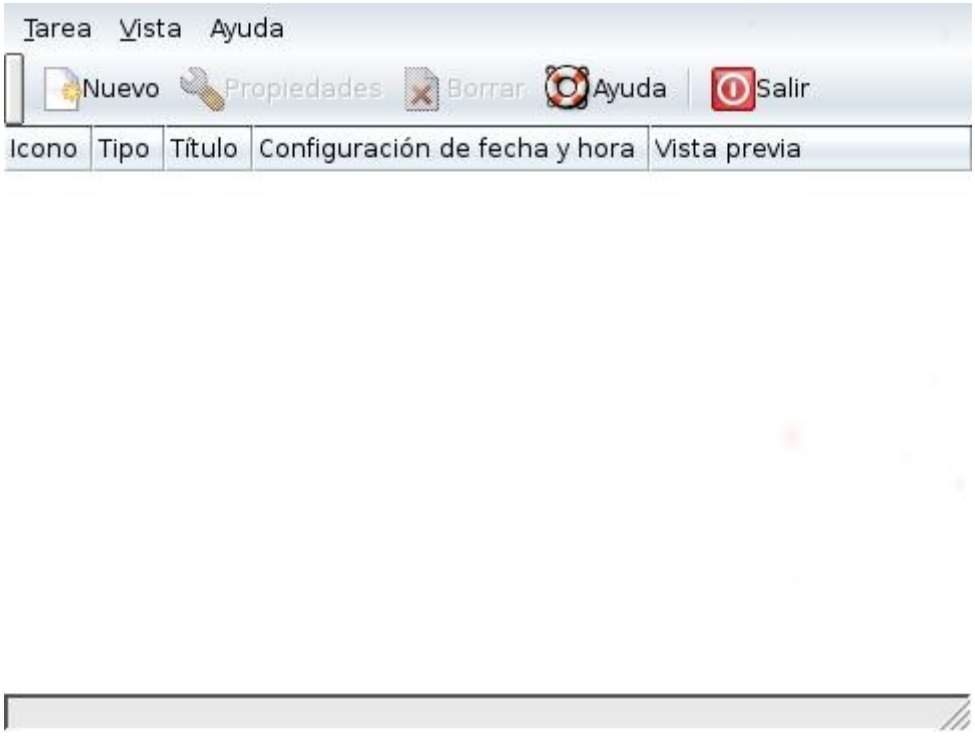

Comentario:
Programa:
ls / > /home/matados2k/lista.txt
Buscar...
Activado
Silencioso
Meses
Enero
Febrero
Marzo
Abril
Mayo
Junio
Julio
Agosto
Septiembre
Octubre
Noviembre
Diciembre
Ajustar Todo
Días del Mes
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
Ajustar Todo
Días de la Semana
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
Ajustar Todo
Diario
Se ejecuta todos los días
Horas
AM
0 1 2 3 4 5
6 7 8 9 10 11
PM
12 13 14 15 16 17
18 19 20 21 22 23
Ajustar Todo
Minutos
0 5 10 15 20 25
30 35 40 45 50 55
Ajustar Todo
Aceptar
Cancelar


Archivo Editar Preferencias Ayuda
Tareas/Variables
Valor
Tareas
ls / > /home/matados2k/lista.txt
Variables
MAILTO
matados2k
Listo.

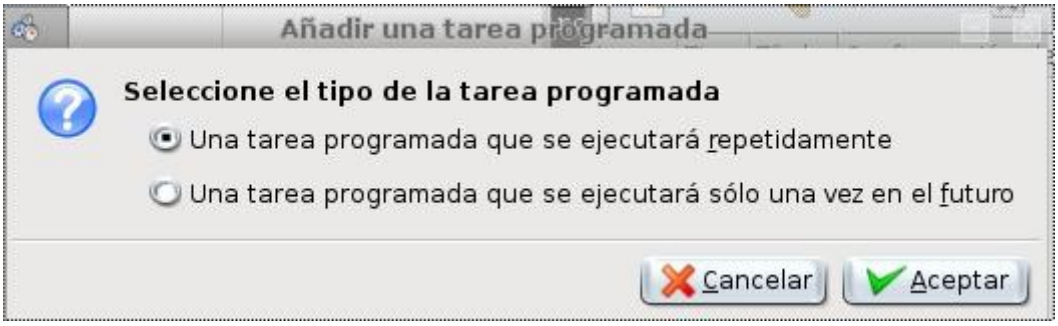
Con esto solo nos queda guardar y listo. Y recordad este programa usa los crontabs propios del usuario.

Gnome-schedule.

Cuando lo abrimos nos encontramos con lo siguiente:



Al darle a nuevo nos dejará elegir entre los dos tipos de programaciones que conocemos:



Si elegimos la que es como 'cron' nos encontramos dos interfaces, una simplificada:

Y otra avanzada:

Para la opción del estilo 'at' se nos muestra la siguiente pantalla:

Y sólo nos queda configurarla en la pestaña de Fecha y hora:

Por desgracia esta aplicación no nos deja configurarvariables, ambas herramientas podrían mejorar incluyendo lo que les falta. También hace uso de los crontabs de usuario.

Despedida.

Si, lo se. La entrega es rápida y corta. Pero es que esto es así de fácil y sencillo. Hasta la semana que viene.