Linux

Linux es un S.O. (Sistema Operativo) Open Source (Código Abierto) existen varias distribuciones de SO que están construidas sobre este núcleo.

- ubuntu
- redhat
- fedora
- debian
- linux mint
- mandriva
- y varias más

Algunas de estas distribuciones y otras que no menciono tienen versiones de pago, pero la mayoría se pueden utilizar libremente.

Suelen tener versiones para instalar en lo que se denomina live-cd que nos permite probar el sistema sin necesidad de instalarlo.

Pueden hacerse instalaciones, con las que conseguimos que en nuestra máquina convivan varios sistemas operativos, se instalará tanto el nuevo sistema operativo como un gestor de arranque, que nos permite mediante un menú de inicio seleccionar el S.O. con el que vamos a trabajar. Este proceso puede ser mas técnico, o automático dependiendo de nuestros conocimientos.

Todas estas versiones tienen métodos muy sencillos para instalar multitud de aplicaciones, que pueden ser una buena alternativa a las aplicaciones basadas en win.

Tipos de Consola de Comandos

En Linux existen varios tipos de consola de comandos (texto) desde las cuales podemos controlar todas las funcionalidades de nuestro equipo, ficheros, seguridad, aplicaciones, etc.

Algunas de las consolas más conocidas y utilizadas son las siguientes:

- bash: está basada en la consola bsh, utilizada en versiones anteriores de UNIX.
- bsh: consola en la que se basa bash. bsh en las nuevas versiones de Linux suele ser un enlace simbólico a bash.
- tcsh: consola evolución de csh. consola popular entre los programadores C.
- csh: consola C.
- ksh: evolución que auna y amplia bsh y csh.
- zsh: evolución y amplia ksh.

Existen muchas más consolas, pero no voy a listarlas todas.

La elección de una u otra es cuestión de costumbres, o necesidades específicas.

Las distribuciones actuales suelen tener un entorno gráfico desde el cual, mediante una opción de menú se suele poder abrir sin problemas una consola, en el caso de no encontrar esta opción de menú, podemos buscar una que nos permita ejecutar un comando, y ejecutaremos `xterm` o `Konsole`

Comandos Internos y Externos

Como en todos los SO existen comandos internos y externos, la diferencia fundamental es que los internos están incorporados a la consola y se pueden ejecutar directamente, mientras que para los externos hay que indicar la ruta hasta la ubicación del comando.

Para los comandos externos puede ser que no tengamos que indicar la ruta hasta la ubicación del mismo de forma explicita, si esta ruta esta incluida en la variable de entorno PATH.

También debemos tener precaución en el caso de que el comando exista tanto de forma interna y externa, ya que las dos versiones del comando pueden dar resultados distintos, por lo que si queremos estar seguros de que estamos ejecutando la versión externa debemos indicar la ruta del comando (p.e. pwd ó /bin/pwd)

Consola - Básico

Algunos conceptos básicos sobre la consola, están explicados de forma muy superficial, pero sirve para ir haciéndose una idea
INTRODUCCIÓN DE COMANDOS

Si empezamos a escribir un comando y antes de terminar podemos utilizar la tecla `tab` para que se nos complete si es posible con el comando mas adecuado (si lo hacemos con un nombre de archivo que pasamos como parámetro también funciona) si existe mas de un posible resultado, si pulsamos nuevamente la tecla `tab` nos mostrará una lista con los posibles resultados.

Existe un historial de comandos utilizados, podemos acceder a él de varias formas, teclas de cursor (??), con `ctrl-R/ctrl-S` para iniciar una búsqueda hacia atrás y hacia adelante respectivamente, podemos acceder a la lista completa con `history` borrarlo con `history -c` (ver la lista completa con `man bash`)

CONFIGURACIÓN DE LA CONSOLA

Ficheros de generales

- `/ etc/bash.bashrc`
- `/ etc/profile`

Ficheros específicos del usuario (~ indica el directorio del usuario `/home/usuario`)

- `~ /.bashrc`
- `~ /.profile`

USO VARIABLES DE ENTORNO

Declarar/asignar variable de entorno

- `VARIABLE=valorVariable`
  `export VARIABLE`
- `export VARIABLE=valorVariable`

Ver valor de una variable de entorno concreta (IMPORTANTE EL &)

- `echo $VARIABLE`

Ver valores de todas las variables de entorno

- `env`

Eliminar variable de entorno

- `unset VARIABLE`

OBTENER AYUDA

Existen varias formas de obtener ayuda de algunos comandos, ficheros, carpetas, ...

- `man palabraClave`
- `info palabraClave`
- `comando -h`

Algunas palabraClave no obtendrá resultados con algún comando de ayuda y otras tendrán con varias, `man` e `info` tienen una explicación completa mientras que los comandos con la opción `-h` es una ayuda mas breve y concisa.

`man` e `info` utilizan `less` para mostrar la información, es un paginador, utilizar man `less` para ver sus opciones y poder "navegar" por el contenido de ayuda

man clasifica la información en secciones (1-9), normalmente nos muestra la información de nivel mas bajo, para forzar que muestre una sección, en el caso de que el comando pertenezca a varias, lo haríamos siguiendo esta sintaxis man X comando (X=número de sección).

nº. sección	Descripción
1	Programas ejecutables / comandos de consola
2	Llamadas de sistema facilitadas por el kernel
3	Llamadas proporcionadas por bibliotecas de programa
4	Ficheros de dispositivo (/dev normalmente)
5	Formatos de fichero
6	Juegos
7	Varios
8	Comandos de administración de sistema (root)
9	Rutinas del Kernel

PROMPT

Nos podemos encontrar varios formatos de prompt o indicador de lincea de comandos, además de varia información que podemos configurar a nuestro gusto al final del lado derecho podremos encontrar:

- $ Indica que la consola se esta ejecutando con permisos de usuario NO root
- # Indica que la consola se esta ejecutando con permisos de usuario root

Flujos, Redirecciones y Pipes - Flujos

En esta sección vamos a ver que es esto de los Flujos, Redirecciones, y Pipes (tuberías o canalizaciones), vamos a ver que son los flujos para poder entender y aprovecharnos de los redireccionamientos y Pipes

Principales tipos de flujos de entrada y de salida:

- **Entrada Estándar (stin)**: Es el flujo estándar desde el que sistema (sistema y programas) acepta la entrada de información, normalmente el teclado.
- **Salida Estándar (stout)**: Es el flujo estándar a través del cual el sistema (y programas) devuelven información, normalmente la consola(o ventana xterm).
- **Error Estándar (stderr)**: Es el flujo estándar a través del cual el sistema (y programas) devuelve información sobre errores, normalmente es el mismo que el de **(stin)**, aunque en algunos casos se suelen redirigir a ficheros, (p.ej. los ficheros de log que recogen errores del sistema), para su estudio posterior. También se puede mantener el mostrar estos mensajes por la salida estándar a la vez que se redirige una copia a un fichero/s.

Flujos, Redirecciones y Pipes - Redirecciones

Al redireccionar un flujo lo que hacemos es modificar el comportamiento natural del sistema o programa, p.ej. enviar la salida de un programa a un fichero, tomar la entrada para un programa de un fichero, un poco mas abajo veremos unos ejemplos.

Veamos una tabla con los operadores más comunes de redirección

Operador Redirección	Efecto del Operador
>	Redirige la stdout a un fichero, si no existe lo crea, y si existe lo sobre escribe (borra el contenido antes de incluir el nuevo contenido)
>>	Añade la stdout a un fichero, pero a diferencia del anterior la añade al fichero especificado
2>	Redirige la stderr a un fichero, si no existe lo crea, y si existe lo sobre escribe (borra el contenido antes de incluir el nuevo contenido)
2>>	Redirige la stderr a un fichero, pero a diferencia del anterior la añade al fichero especificado

Operador Redirección	Efecto del Operador
`&>`	Crea un fichero tanto con stdout como con stderr , si ya existe elimina primero su contenido
`<`	Envía el contenido del fichero especificado como stdin
`<<`	Utiliza como stdin las líneas de texto pasadas
`<>`	El fichero especificado será utilizado tanto como stdin como stdout

Estos direccionamientos se pueden utilizar de diversas formas con algunos conocimientos y algo de imaginación pueden a llegar a ser una herramienta muy potente.

- Podemos redireccionar stderr para que no se muestren los mensajes de error al ejecutar un programa determinado
  - `programa 2>/dev/null`
  - /dev/null es un dispositivo que apunta a null haciendo que se pierdan en este caso las salidas de error.

- Podemos utilizar << dentro de un script para hacer que acepte datos desde la línea de comandos hasta que encuentre una línea con la cadena EOF
  - `programa<<EOF`

tee

El comando tee nos permite enviar la stdin tanto a stdout como a aquellos ficheros que deseemos, con ello podríamos mostrar el resultado de un programa tanto por la pantalla como almacenarlo en un fichero.

p.ej.:

```
programa | tee ficheroSalida.txt
```

Vemos que con "|" canalizamos la salida de programa a "tee" y con este ultimo hacemos que dicha salida aparezca tanto por stdout como al fichero ficheroSalida.txt

Recordad que con man o info desde la consola podemos ver todas las opciones de los distintos comandos que podamos necesitar

Flujos, Redirecciones y Pipes - Pipes (Canalizaciones)

Pasar datos entre programas

Las Pipes (canalización) se usan para redireccionar la stdout de un programa hacia la la stdin de otro, o lo que es lo mismo utilizar la salida de un programa como entrada de otro, en principio no hay límite en el número de programas utilizados, veamos un ejemplo para genérico para comprender su funcionamiento

```
programa1 | programa2 | script3 | ..... | programaN
```

Ahora un ejemplo mas práctico, el cual utilizaremos para ver la línea de la salida de ifconfig (herramienta que nos devuelve la información de una tarjeta de red) en la que aparece la MAC de la misma, mediante grep

```
ifconfig eth0 | grep direcciónHW
```

la cual, suponiendo la existencia en nuestro sistema de una tarjeta de red eth0, nos devuelve algo parecido a:

```
eth0 Link encap:Ethernet direcciónHW 00:15:f2:1b:65:ec
```

Las Pipes pueden llegar a ser una herramienta muy potente, mucho mas de lo que parece con la primera impresión, tanto para seguimiento de errores, como en configuración del sistema, seguridad,... y todo aquello que la imaginación nos permita

Pipes y Comandos - Comandos

Vamos a ver un ejemplo en el que utilizaremos la salida de un comando mediante una Pipes (|), para utilizarla con otro comando para realizar una tarea repetitiva, y así agilizar nuestro trabajo.

Voy a utilizar algunos comandos que no hemos visto, que veremos mas adelante en una sección dedicada a los comandos mas comunes, por lo que simplemente diré lo que hacen a forma de indicación. Si alguien tiene interés en ir investigando para conocer más a fondo los distintos comandos, os recuerdo que podéis utilizar man (`man comando`) o info (`info comando`) para ver todas las opciones y algunos ejemplos de los comandos que vayamos viendo.

Abriremos un terminal si estamos utilizando una distribución con entorno gráfico, que en estos tiempos será lo mas habitual.

1. Vamos al menú principal, en mi caso ubuntu, y buscamos:
    - `menú principal->aplicaciones->accesorios->terminal`
2. Creamos un directorio para realizar nuestros ejemplos, con seguridad para no modificar ningún archivo de nuestro sistema, de forma accidental, evito utilizar nombres con espacios y caracteres especiales:
    - `mkdir ejemplos-lt`
3. Cambiamos nuestra ubicación dentro de nuestro nuevo directorio:
    - `cd ejemplos-lt`
4. Creamos unos cuantos ficheros, `touch` nos permite hacer esto:
    - `touch 1.111 1.112 1.113 1.121 1.131 1.141`
5. Comprobamos que efectivamente se han creado 6 archivos:
    - `ls`
6. Como podéis ver hemos creado 6 archivos, ahora vamos a borrar solo los que acaban en "1", esto lo podríamos hacer simplemente con el comando `rm` pero lo vamos a complicar solamente con el fin de ver como podemos utilizar pipes, desde luego que para esto no es necesario utilizar pipes, pero con

algo mas de conocimientos y de imaginación seguro que en el futuro se nos ocurren muchos casos en el que utilizarlas:

7. Listamos solamente los archivos acabados en "1", y vemos que hay 4:
    - `ls *1`
8. Eliminamos los archivos utilizando un pipe (|):
    - `ls *1 | xargs rm`

Con el pipe le pasamos a `rm` mediante `xargs` cada resultado devuelto por `ls *1`, por lo que se construyen 4 comandos `rm` y el nombre de cada fichero que cumple el patrón `*1`

En este caso en concreto podríamos haber realizado lo mismo con:

```
rm *1
```

Pero como indiqué antes hemos utilizado un pipe para ilustrar un ejemplo sencillo

Trabajando con Ficheros

Los comandos que vamos a ver normalmente modifican los contenidos enviando el resultado a stdout, no modificando realmente el fichero original, aunque podemos redirigir o canalizar esta salida a un fichero o comando.

Comandos para combinar ficheros

Vamos a ver algunos comandos para combinar ficheros

cat

Concatena, o une el contenido, ficheros y envía el resultado a la stdout la cual podemos redireccionar a un fichero

- Concatenación de dos ficheros y redireccionar el resultado a un fichero, si el fichero concatenado existiera previamente su contenido se perdería
    - `cat fichero1 fichero2 > ficheroConcatenado`
- Concatenación de dos ficheros y redireccionar el resultado a un fichero, si el fichero concatenado existiera previamente su contenido NO se perdería
    - `cat fichero1 fichero2 >> ficheroConcatenado`
- Ver el fichero resultante, `cat` concatenará `ficheroConcatenado` con nada y el resultado lo dirige a `stdout`
    - `cat ficheroConcatenado`
- Crear contenido en un fichero, si el fichero existía su contenido se perderá. Tras ejecutar el siguiente comando, escribir el texto deseado, podremos utilizar `return` para crear saltos de línea, para terminar de introducir contenido pulsar `ctrl+d`
    - `cat > fichero`
- Añadir contenido a un fichero, si el fichero existía, el nuevo contenido se añadirá al final del contenido existente. Ejecutar el siguiente comando y seguir las directrices del comando anterior

- `cat >> fichero`

`cat` cuenta con algunas opciones y modificadores, los cuales alteran los resultados producidos, como añadir números de línea, mostrar caracteres especiales,... como siempre utilizar `man` o `info` para ver la lista completa y su sintaxis

tac

Todas las explicaciones para `cat` son válidas para `tac` pero con el orden de las líneas invertido en los resultados.

join

Une dos ficheros basándose en un campo, por defecto el primero, enviando el resultado a stdout, los dos ficheros deben estar ordenados correctamente para poder realizar una unión completa, de no ser así join realizará la unión hasta que encuentre la primera línea desordenada.

Los campos vienen delimitados por un carácter predeterminado, por defecto un espacio en blanco " ", y cada línea representa un registro, con lo que podemos tener una pequeña base de datos, con una funcionalidad limitada, pero que como práctica es interesante.

En el directorio creado anteriormente para nuestros ejemplos, creamos dos ficheros con dos campos cada uno y los siguientes contenidos, para cada uno de ellos, lo podemos hacer con cat y así practicar, o utilizar un editor GUI de nuestra distribución

fichero: uno

1 uno1
2 dos1
3 tres1
4 cuatro1

fichero: dos

1 uno2
1 uno21
2 dos2
3 tres2
4 cuatro2

unimos los dos ficheros y lo vemos en consola, fijémonos en el resultado y como maneja las entradas repetidas del segundo fichero

```
join uno dos
```

si queremos guardar el resultado utilizaremos una redirección, por ejemplo creando un nuevo fichero o reemplazando su contenido completamente

```
join uno dos > ficheroJoin
```

paste

Une dos ficheros línea a línea separándolas con una tabulación, experimenta por tu cuenta.

Comandos para la transformación de ficheros

expand

Cambia tabulaciones por espacios en blanco, puede ir muy bien para por ejemplo, modificar ficheros de código de programación, en el que se han utilizado tabulaciones para indentar el código, en estos casos con pocos anidamientos, al imprimir el código queda muy mal y se dificulta su lectura, por lo que podemos cambiar las tabulaciones por dos espacios en blanco, mejorando así su lectura/estudio.

unexpand

Lo contrario que `expand`

od

Permite ver ficheros en formato octal, aunque aplicando algunas opciones podemos verlos en formato hexadecimal, y otros atendiendo a la longitud de palabra,...

sort

Ordena el contenido del fichero/s, normalmente utiliza el primer campo y los valores ASCII, aunque tiene opciones que permiten modificar este comportamiento.

split

Permite dividir un fichero en base a unas condiciones establecidas mediante sus opciones, DEBEMOS pasarle el prefijo para los ficheros resultantes, a dicho prefijo le añadirá un conteo alfabético a cada fichero resultante, (p.e. ficheroa,ficherob,ficheroc,...). Dependiendo de las opciones elegidas podemos dividir el fichero por bytes, líneas, ...

tr

Cambia uno a uno los caracteres que se encuentran en el grupo1 por los del grupo2 en la stdin y recordemos que podemos utilizar la redirección para realizarlo sobre un fichero, y mostrar el resultado en pantalla o a un fichero.

uniq

Elimina líneas duplicadas, útil después de ordenar un fichero con `sort` y no se quiera tener entradas duplicadas.

fmt

Principalmente formatea el ancho de las líneas de la stdin o de un fichero.

nl

Herramienta para numerar líneas, por ejemplo de un fichero de código. Tiene varias opciones interesantes.

pr

Prepara un texto para su impresión, ancho de línea, líneas por página, ...

Comandos para la visualización de ficheros

head

Muestra el inicio de un fichero, por defecto las 10 primeras líneas, tiene algunas opciones.

tail

Muestra el final de un fichero, por defecto las 10 ultimas líneas, tiene algunas opciones. Es especialmente útil para ver las últimas entradas de los ficheros de log, tiene opciones muy interesantes, que nos permite hacer el seguimiento de un fichero, o que lo haga hasta que termine un proceso determinado.

more

Visualizar de forma paginada un fichero

less

Visualizar de forma paginada un fichero, versión mejorada de `more` que nos permite ir tanto hacia adelante como hacia atrás, buscar en ambos sentidos, ir a una línea especifica, ...

Comandos para resumir ficheros

cut

Corta las líneas entrada según lo especificado en sus opciones, y lo muestra por la salida (recordad que podemos utilizar redirecciones y utilizar ficheros)

```
cut -b2-10
```

escribir y terminar con `ctrl+d`

de cada línea devolverá desde el carácter 2 al 10, empieza a contar desde 1

wc

Cuenta las palabras, líneas, bytes,... de un fichero.

Expresiones regulares

Las expresiones regulares son cadenas que se utilizan como patrones para comparaciones

- La forma mas simple es un texto, p.e. `txt` que coincidirá con todas las cadenas que sean exactamente txt o aquellas que la contengan.

- Podemos hacer que coincidan con uno varios caracteres, colocándolos entre corchetes([]) `[aeiou]`
- Coincide en un rango `[a-z]`
- Un carácter `.` p.e. `r.cord` coincidirá con cualquier cadena de 3 caracteres que empiece por "r" le siga cualquier carácter y acabe con "cord" (record, racord)
- Para indicar que la cadena debe iniciarse como indica la expresión regular, utilizaremos el `^`
- Para indicar que la cadena debe terminar como indica la expresión regular, utilizaremos el `$`
- Repetición (podemos combinarlas una combinación muy útil es `.*` que representa cualquier subcadena)
    - `?` indica cero o una aparición de la expresión regular
    - `+` indica una o mas apariciones
    - `*` cero o mas apariciones
- Una cadena u otra `|` se puede utilizar para un reemplazo como `silla|taburete` para cambiarlo por `asiento`
- Para especificar como se aplican los modificadores/operadores utilizamos los `()` p.e. `^(silla|taburete)s`
- Si queremos utilizar alguno de los operadores propios de las expresiones regulares dentro de la cadena precederemos dicho carácter con `\` p.e. `^\(((silla|taburete)s`

Existen grandes tratados sobre expresiones regulares, así que no me extiendo mas, al menos de momento, seguro que las iremos utilizando en ejemplos.

grep

Potente herramienta para buscar coincidencias en ficheros, nos permite utilizar expresiones regulares, canalizaciones, redirecciones, recomendable dedicarle un tiempo a practicar con el y adquirir soltura.

sed

Herramienta para modificar la entrada y muestra el resultado en la stdout, se pueden utilizar redirecciones como es normal, es una herramienta compleja ver todas las opciones que nos proporciona con `man` o/y `info`

utilizar `man` o `info` para ver todas las opciones de los distintos comandos vistos

Gestionar Software - Paquetes

Paquetes ¿Qué son?

Los paquetes como su nombre indica son paquetes de software que incluyen los ficheros, en algunos casos muchos, que componen una aplicación. Estos paquetes pueden incluir ficheros fuente o bien binarios, numero de versión, así como las referencias a las librerías comunes que necesite, ahora explicaré que significa cada cosa.

Ficheros fuente, son aquellos escritos directamente por los desarrolladores de la aplicación, si utilizamos

estos paquetes tenemos algunas ventajas e inconvenientes. Empezaré por los inconvenientes, debemos compilar nosotros mismos el software, con lo que debemos saber hacerlo, y hacerlo bien, ya que si lo hacemos de forma descuidada, o errónea podríamos entre otras cosas generar una aplicación con permisos de root lo que le concedería acceso total a nuestro sistema lo que podría ser un agujero de seguridad muy importante. Como ventaja es que la aplicación estaría adaptada totalmente a nuestro sistema, pudiendo así aprovechar los recursos del mismo plenamente, otra ventaja es que al tener acceso al código fuente, si tenemos ganas y paciencia podremos comprobar que la aplicación no realice acciones indeseadas (p.e. acceso a información confidencial).

Ficheros binarios, son aquellos compilados con unos valores genéricos, lo que les hace aptos u operativos para una arquitectura concreta o varias (tipos de procesares, familia Intel i386 X64, Sparc, powerPC, ...). Los inconvenientes principales son una posible disminución en el aprovechamiento de los recursos del sistema, otro inconveniente seria que si el paquete proviene de una fuente desconocida, no podremos comprobar si realiza alguna acción no deseada. Como ventaja principal la comodidad, prácticamente no tendremos que hacer nada para utilizar la aplicación.

Librerías comunes, en las aplicaciones hay tareas repetitivas que pueden ser utilizadas por varias aplicaciones, minimizar una ventana, barras de desplazamiento, gestión de memoria, operaciones en coma flotante, etc. Pues bien los ficheros que se encargan de realizar todas estas tareas podrían estar incluidas en la propia aplicación, lo que nos llevaría en el caso de tener funcionando varias aplicaciones que utilicen estas tareas, a unas necesidades de memoria desorbitadas, por lo que se utilizan unas librerías compartidas que realizan estas tareas y que se van cargando si no están cargadas ya por otra aplicación en memoria según se van necesitando. aparte de los recursos que liberamos con esta técnica, los desarrolladores evitan tener que reprogramar cada vez que necesiten de estas funcionalidades, dependiendo de la arquitectura, aplicación, etc.

Estas **librerías comunes**, o **funcionalidades compartidas** se conocen como **dependencias**, lo que viene a querer decir que la aplicación del paquete que queremos instalar necesita ciertas librerías para poder funcionar, ya que va a intentar utilizar las funcionalidades ofrecidas por estas.

Gestores de Paquetes

Para instalar las aplicaciones, podemos copiar los archivos que la componen en los directorios adecuados, buscar las dependencias en la versión adecuada, ver si ya están instaladas en nuestro sistema, en las rutas que la nueva aplicación espera, si no lo estan hacerlo ....,

Por el contrario si lo que queremos es desinstalar alguna aplicación, podemos buscar su ubicación, la de los archivos de configuración, las dependencias y comprobar que estas nos son utilizadas por otra aplicación, y eliminarlo todo.

O bien podemos utilizar un gestor de paquetes que realizara todas estas tareas, y otras que no he comentado por abreviar, por nosotros.

Cada distribución utiliza un gestor especifico, que es el que suele venir por defecto, aunque si nos gusta cualquier otro normalmente funcionará, ahora bien, aunque si no es estrictamente necesario, lo recomendable es utilizar el que la distribución aconseja. Estos gestores de paquetes nos facilitan la instalación, actualización y eliminación de software, manteniendo una base de datos en la que se refleja el el software instalado, versiones, dependencias,...

Aunque los gestores de paquetes pueden funcionar en distribuciones en las que no vienen instalados por defecto, hemos de tener en cuenta que cada uno tiene sus peculiaridades, por lo que en un sistema con varios gestores de paquetes, podemos instalar varias veces el mismo software (aplicaciones, bibliotecas ...) una vez por cada gestor de paquetes, ya que no comparten las bases de datos, y ocasionar inestabilidades en el

sistema e incluso su colapso y quedar inutilizable. Dicho esto si decidimos utilizar varios gestores de paquetes debemos ser muy cuidadosos a la hora de realizar las acciones que nos permiten.

Cada gestor de paquetes tiene sus propios ficheros de configuración en los cuales entre algunas cosas mas se guarda los lugares donde buscar el software, llamados repositorios, aunque algunos como yum buscara en varias ubicaciones intentando localizar el paquete solicitado

Gestores de Paquetes Comunes

RPM Gestor de paquetes utilizado por las distribuciones derivadas de Red Hat, es uno de los mas extendidos. podéis obtener información de su uso desde su web [1]

YUM Gestor de paquetes de Yellow Dog y que utilizan algunas distribuciones mas. Información sobre este gestor en español en esta wiki de fedora project sobre YUM. [2]

DPKG y APT para Debian y sus distribuciones derivadas, mas información en esta wiki de Debian, para APT [3]

Como yo utilizo Ubuntu que es una distribución derivada de Debian explicaré como realizar alguna de las tareas mas habituales, que podemos realizar con APT, aunque en la actualidad existen interfaces gráficas mucho mas comodas de utilizar y normalmente es lo que se utiliza. En Ubuntu lo llaman Centro de Software Ubuntu, y habitualmente tiene una entrada en el menú principal, o lanzador (en unity), si nos decantamos por esta opción, que será lo mas probable, tendremos una larga lista de aplicaciones para instalar.

Si queremos instalar una aplicación que no aparece en la lista inicial, será suficiente con buscar en google y añadir el repositorio a la lista de lugares donde buscar aplicaciones, se puede hacer desde el entorno gráfico o bien siguiendo las instrucciones que suele poner en la web de los desarrolladores, debemos tener en cuenta que solo debemos incluir repositorios que merezcan nuestra confianza, ¿como...?, efectivamente buscamos en google a ver si ese repositorio tiene entradas en internet que digan algo al respecto

apt

apt es una aplicación importante, por lo que debemos ejecutarlo como un usuario con los permisos necesarios.

Instalar un paquete (normalmente tendrá en cuenta sus dependencias)

```
sudo apt-get install nombre-de-paquete
```

Eliminar un paquete

```
sudo apt-get remove nombre-de-paquete
```

Eliminar ficheros de paquetes eliminados y los de configuración

```
sudo apt-get purge
```

Existen mas opciones interesantes, algunas nos permiten descargar los paquetes fuentes, compilarlos, instalarlos y tener en cuenta/forzar la satisfacción de sus dependencias

Utilizar paquetes de una distribución distinta - alien

Cada paquete esta construido para su distribución, o dicho de otro modo, cada aplicación debe tener un paquete construido para cada gestor con el que pueda ser utilizado, si no existe podemos construirlo a partir del los ficheros fuente.

Si tenemos un paquete para una distribución y queremos utilizarlo en otra para el que no existe paquete, podemos utilizar una aplicación llamada `alien`, con una simple busqueda en google encontrareis varios tutoriales sobre el mismo (ademas de man, info, apropos).

Bibliotecas Compartidas

Bibliotecas Compartidas - ¿Qué son?

Las Bibliotecas compartidas son ficheros que contienen colecciones de fragmentos de código, que nos permite el poder utilizarlos en el nuestro código o/y por los programas que tenemos instalados en nuestra máquina, sin tener que volver a programar ciertas tareas rutinarias (p.e. crear ventanas, botones, acceso a dispositivos, ...).

Además de ahorrarnos tiempo a la hora de desarrollar aplicaciones, y posibles errores, estas bibliotecas al ser compartidas se instalan en nuestra máquina y como su nombre indica son o pueden ser compartidas por varios programas, consiguiendo así:

- Ahorrar espacio de almacenamiento
- Se cargan una sola vez en memoria, independientemente de las aplicaciones que las requieran

Cada aplicación suele utilizar una versión concreta de cada biblioteca compartida, pero esto en Linux no es ningún problema ya que Linux utiliza un sistema de numeración de versiones, que posibilita el tener instaladas varias versiones de cada biblioteca.

Las aplicaciones deben poder encontrar las bibliotecas, por lo que los cambios deben ser reflejados en unos ficheros de configuración, que son los que las aplicaciones consultan para poder localizarlas.

Si una biblioteca deja de estar accesible, las aplicaciones fallaran, o incluso lo hará el sistema completo.

Todo esto nos lleva a que debemos tener un buen control de nuestras bibliotecas, que normalmente no nos causará problemas si utilizamos el gestor de paquetes correspondiente a nuestra distribución (y no utilizamos opciones que ignoren dependencias).

Los nombres de las bibliotecas compartidas en Linux terminan en .so (shared object, objeto compartido).

Rutas de los ficheros de Biblioteca

Algunas de las rutas son tenidas en cuenta por Linux aunque no las especifiquemos como `/lib` y `/usr/lib`

Existen varias formas de definir la ruta a los ficheros de biblioteca.

Editando el fichero `/ etc/ld.so.conf` . En este fichero se encuentran algunas de las rutas a bibliotecas, una por línea.

En `/ etc/ld.so.conf` pueden existir líneas `include` que como saben los programadores de C indica que se debe incluir el contenido de otro/s fichero/s, p.e. en Ubuntu contiene una línea `include / etc/ld.so.conf.d/*.conf` que indica que se deben incluir todos los ficheros que terminen en .conf dentro del directorio `/ etc/ ld.so.conf.d/` y estos a su vez pueden contener otras líneas `include`. En otras distribuciones se adoptan soluciones parecidas aunque con algunas pequeñas diferencias.

Linux en una de sus tareas de arranque lee todos los ficheros que puedan estar involucrados en la definición de rutas a bibliotecas (incluidas aquellas que tiene siempre `/lib` `/usr/lib`). y genera un sistema de datos mas optimizado que los ficheros en texto plano que se utilizan para configurar las rutas, por eso cuando modifiquemos un fichero de en el que se encuentren rutas a bibliotecas debemos ejecutar la herramienta `ldconfig` que se encarga de leer los ficheros de configuración y volver a generar este sistema de datos optimizado con las rutas de las bibliotecas.

Normalmente no tendremos que realizar esta tarea, ya que utilizaremos el gestor de paquetes de nuestra distribución el cual, normalmente, realiza todas las acciones necesarias al instalar un nuevo paquete. Estas tareas solamente nos veremos obligados a hacerlas cuando instalemos alguna biblioteca compilada por nosotros mismos, o en circunstancias poco habituales.

Cambiar la Ruta de Bibliotecas de Forma Temporal

Imaginemos que queremos utilizar temporalmente una biblioteca, p.e. para comprobar su efecto en el sistema, podemos guardarla en una carpeta que tenemos para estos temas quizá ~/pruebas/lib (recordad que ~ hace referencia a la carpeta home de usuario). Ahora que ya tenemos localizada la ruta para probar nuestra librería, podemos utilizar la variable de entorno `LD_LIBRARY_PATH` las rutas añadidas de este modo serán colocadas al principio de la cadena de búsqueda interna para las librerías por lo que siempre se utilizarán antes que las que se estaban utilizando hasta el momento, si es que sustituye a alguna.

El comando a utilizar sería

```
export LD_LIBRARY_PATH=~/pruebas/lib/libreria1:~/pruebas/lib/libreria2
```

Como podemos ver se pueden poner tantas rutas como necesitemos separadas por ":" , no tienen porque estar en la carpeta de usuario, en principio somos libres de utilizar la ruta que necesitemos, siempre y cuando tengamos permiso para acceder a ella.

Para esta tarea no es necesario ejecutar `ldconfig` para actualizar las rutas.

Detectar y Corregir Problemas con las Rutas de las Bibliotecas

Si estamos utilizando un entorno gráfico, si algún programa no se ejecuta por problemas con las librerías, normalmente no veremos la razón del fallo. Lo podremos averiguar de varias formas pero normalmente en nuestra consola, podemos escribir el nombre el programa para ejecutarlo y si se produce un error por la falta de algunas librerías nos devolverá un mensaje indicándonos que librerías nos faltan, o que su ruta no está registrada correctamente.

Supongamos que un programa nos devuelve un error indicándonos que no encuentra la librería "libreriaQueFalta.so", si normalmente utilizamos el gestor de paquetes, esto no debería suceder.

Primero buscaríamos el fichero a ver si esta instalado, p.e. podemos utilizar `find` y si no lo encontramos buscaremos el paquete al que pertenece y lo instalaremos con el gestor de paquetes.

Si el fichero de biblioteca lo tenemos instalado, probablemente el problema esté en que no se encuentra en la ruta que el programa espera, podremos averiguar cual es esta ruta que el programa espera ejecutando ldd /ruta/al/programa/programa , también puede suceder que tengamos instalada una revisión de la versión que no es exactamente la que el programa espera, si esta es una revisión menor p.e. tenemos la versión libreriaQueNecesitamos.so.3 y la que tenemos es la libreriaQueNecesitamos.so.3.4, por norma estas revisiones solamente tienen correcciones menores, por lo que podremos utilizarla.

Tanto si nuestro problema es que la ubicación no es la que espera el programa como si es que la versión de la que podemos disponer es una revisión menor (siempre si la que tenemos es superior a la esperada) la solución es la misma:

Crear un enlace simbólico como root y encontrándonos en el directorio donde tengamos guardada la biblioteca

```
sudo ln -s libreriaQueNecesitamos.so.3.4 libreriaQueNecesitamos.so.3
```

ó en su caso desde la carpeta donde se espera que este la biblioteca utilizando la ruta completa de la biblioteca original

```
sudo ln -s /ruta/a/la/libreria/instalada/libreriaQueNecesitamos.so
libreriaQueNecesitamos.so
```

Tras esto debemos ejecutar `ldconfig` para que se actualicen las rutas en el sistema.

Administrar Procesos

En este apartado veremos como administrar procesos (identificación de procesos la manipulación de procesos de primer y segundo plano, destrucción de procesos, regulación de prioridades,..).

¿Qué es un Proceso? ¿Y Una Tarea?

Cuando ejecutamos un programa este desencadena o puede desencadenar una serie de tareas, las cuales normalmente a su vez pueden desencadenar tareas y/o procesos. Lo vemos con un ejemplo muy sencillo

Imaginemos que ejecutamos un programa de facturación, pues este desencadenará, o mas bien podrá hacerlo, varias tareas, en un momento dado querremos imprimir una factura, y ... ahí está, se lanzará una tarea (la tarea de impresión), la cual disparará a su vez varios procesos (o tareas, dependiendo de la complejidad), como por ejemplo, comprobar estado de impresora, inicializarla, ...

Podríamos decir que una tarea esta compuesta por uno o mas procesos, mientras que el proceso es un fragmento de código que no podemos ejecutar en varias acciones (no es la definición mas estricta pero si es sencilla de comprender).

El Kernel

El Kernel o núcleo del sistema es el proceso principal, una vez iniciado poco podremos administrar de el, a no ser que lo volvamos a reiniciar (reiniciando el ordenador).

Podremos obtener cierta información sobre el kernel con `uname -a` este comando tiene varias opciones

pero `-a` muestra toda la información disponible, simplemente ejecutarlo en una consola y ver el resultado

Ver las Listas de Procesos - ps

ps

ps es un comando con el que podremos ver una lista de procesos, junto con cierta información relativa a dichos procesos. La información devuelta por ps varia dependiendo de las opciones utilizadas,

Podremos utilizar tres tipos de opciones con `ps`:

- **Unix98**: opciones de un solo carácter que se pueden agrupar y deben ir precedidas de un único guión (-).
- **BSD**: opciones de un solo carácter que se pueden agrupar y NO VAN precedidas de un único guión (-).
- **GNU Largas**: opciones multi-carácter no se pueden agrupar y cada una de ellas va precedida de dos guiones (--).

Para cambiar el comportamiento por defecto de `ps` , podemos utilizar la variable de entorno `PS_PERSONALITY`

Con `man` encontraremos una lista completa tanto de las opciones como de los posibles valores de `PS_PERSONALITY`

Por defecto `ps` muestra solo los procesos iniciados desde la terminal desde la cual se ejecuta. Deberemos experimentar con `ps` y sus opciones para ver cual se adecúa mas a las necesidades de cada momento.

Algunas opciones interesantes son:

- **Todos los procesos** `-a -e x`
- **Información adicional** `-f -l j l u v`
- **Jerarquía de procesos** `-H -f --forest`
- **Mostrar una salida superior a 80 columnas** `-w w`, normalmente ps trunca, o corta, la información a 80 columnas. Muy util si redireccionamos la salida, bien a un fichero o a otro comando, ya que podremos utilizar la información completa(imagina un programa que utiliza la ruta del fichero y esta se encuentra truncada, seguro que no obtenemos los resultados deseados)
- **Procesos de un usuario** `-u usuario`, `U usuario`, `--user usuario`

Hay que tener en cuanta que algunas de las opciones se utilizan con -. -- o sin guión dependiendo del tipo de opción a utilizar.

Recomiendo el practicar con este comando e intentar estudiar los distintos resultados, para asi familiarizarse con las distintas opciones y resultados que se obtienen.

La primera línea del resultado de `ps` es un encabezado que nos indica el significado de cada columna, con `man` conseguiremos una lista completa y asi poder interpretar el significado de cada columna.

Para poder ver las lineas referentes a un poceso p.e. synaptiks podemos utilizar la herramienta grep y un pipe (|) `ps -e | grep synaptiks`

Ver las Listas de Procesos - top

top

Es una herramienta que nos muestra información sobre el estado de los procesos que se estan ejecutando el ordenador, es una herramienta parecida a `ps` pero interactiva.

Cuando ejecutamos `top`, nos muestra un listado estructurado en tres zonas, de arriba-abajo nos encontramos con una zona que nos informa de los procesos en ejecución y su estado, el uso de la cpu según tipos de procesos, y el uso de memoria, para poder identificar a que se refiere cada dato utilizaremos `man`. Una información importante es la carga media (load average) que nos indica la carga actual, y las dos anteriores a esta. Estos valores indican un sistema con pocos requerimientos de cpu con un valor de cero y un sistema con un uso intensivo de cpu con un valor de uno. Sistemas con una carga habitual entre 0 y 1 que pase a una carga media superior a 1 podría indicar que tiene problemas con algunos procesos que se encuentren compitiendo por el uso de cpu.

En la siguiente zona encontramos un encabezado que nos indica el tipo de dato que se muestra en las columnas de la tercera zona, para una descripción completa utilizaremos `man`.

`top` acepta opciones de línea de comando que podemos utilizar en el momento de ejecutarlo, tales como el tiempo de retardo de la actualización (-d retardo), procesos específicos hasta un máximo de 20 remitiéndolo (-p pid).

top también soporta comandos interactivos como `q` que sale de `top`, `intro` o `espacio` que refresca la información mostrada, una lista completa de estos comandos interactivos con `man`.

Podremos utilizar top para localizar procesos colgados o zombies y si fuese necesarios destruirlos utilizando los comandos interactivos.

Con `uptime` también podremos averiguar la carga media del sistema

Los pid de procesos para su seguimiento específico con `top` podremos averiguarlos también con `ps`

Ver los Procesos Asociados a una Consola - jobs

jobs

El comando `jobs` nos devuelve los ID de tarea de los procesos lanzados desde la consola en el que se ejecuta, empiezan en 1 y no debemos confundirlos con los PID. `jobs` dispone de algunas opciones, para poder verlas y una breve explicación, debemos utilizar `help jobs`, ya que con man e info no obtendremos información alguna.

Un uso práctico puede ser el asegurarnos antes de cerrar una consola, local o remota, de que no nos dejamos ningún proceso iniciado (bien esté en ejecución o detenido) por temas de seguridad o bien que el proceso se cuelgue.

Si queréis hacer una prueba,

- iniciar un terminal gráfico en este caso y ejecutar `gimp &` (& hace que se ejecute gimp en segundo plano o background y asi poder ejecutar `jobs` desde el mismo terminal).
- Una vez se inicie gimp y nos vuelva a mostrar el prompt del terminal ejecutar `jobs`, (sin cerrar gimp)

Ejecutar y/o alternar procesos entre primer y segundo plano - fg - bg - &

En linux podemos ejecutar procesos en primer plano (foreground) o bien en segundo plano (background).

Un programa en foreground lanzado desde un terminal monopoliza dicho terminal, por lo que en principio, no podremos ejecutar ningún otro programa a la vez (veremos mas adelante como se puede hacer).

Por el contrario un programa en background una vez iniciado, deja de monopolizar el terminal desde el que se lanzo, y este nos vuelve a mostrar el prompt.

¿Cuando lanzaremos un programa en background?

P.e. en un terminal gráfico lanzamos gimp y queremos realizar otras operaciones desde el mismo terminal, o bien vamos a lanzar un programa que no necesita interacción con el usuario (en este ultimo caso nos da igual que sea un xterm o un terminal de texto.

¿Cuando lanzaremos un programa en foreground?

Con un proceso que necesita interacción con el usuario, y esta interacción se realiza a través del terminal.

¿Como podemos lanzar otro programa desde un terminal con otro programa en ejecución en foreground?

Pulsamos `CTRL-z` con lo que pausamos el programa en ejecución y foreground, ojo lo pausamos con lo cual dejará de funcionar, y ya podremos lanzar otro programa p.e. `ls`

- Podemos hacer una prueba lanzamos gimp y comprobamos que podemos operar con el, luego pulsamos CTRL-z y vemos como dejamos de poder trabajar con gimp).

Ahora queremos volver a poner en funcionamiento a gimp y así poder volver a utilizar gimp

- Si queremos devolverlo a foreground escribiremos `fg`.
- Si queremos devolverlo a background escribiremos `bg` (esta sería la opción mas lógica)

En el caso de que tengamos mas de un programa detenido deberemos indicarle tanto a `fg` como a `bg` el ID

de tarea sobre el que actuarán, este ID podemos obtenerlo con `jobs` que hemos visto en un apartado anterior

¿Como lanzar un programa directamente en background - &?

Siguiendo nuestro ejemplo con gimp seria `gimp &` . El & le indica a S.O. que ejecute el programa en segundo plano

Administrar las prioridades de los procesos - nice y renice

¿Qué es la prioridad del proceso?

La prioridad de proceso, se utiliza para decidir la cantidad de tiempo que el proceso podrá utilizar el procesador, por intervalo de tiempo. Paso a explicarlo, el/los procesadores son compartidos por varios procesos (los procesos van alternándose en el uso del o de los procesadores) dando la sensación al usuario que todas las aplicaciones, tareas, procesos se ejecutan a la vez, pues bien la prioridad le dice al sistema que procesos pueden utilizar mas tiempo de procesador y que procesos pasan a un segundo lugar. Esto puede llegar a ocasionar que la ejecución de algún/os proceso/s no llegue/n a ejecutarse nunca, ya que van siendo desplazados en la cola de procesos hacia el final por otros procesos con una prioridad mayor.

Mayor prioridad -20 (menos veinte)

Menor prioridad 19(diecinueve)

Si iniciamos un programa normalmente, y no hay ninguna configuración para el usuario o grupo que lo modifique, este se iniciará con prioridad 0 (cero)

nice

`nice` asigna una prioridad concreta a un programa al ser ejecutado, y por herencia las tareas y procesos que este programa pueda desencadenar.

sintaxis de `nice`

```
nice [argumento] [comando [argumentos-del-comando]]
```

Los argumentos se pueden pasar de 3 formas

- prioridad precedida de un - (a las prioridades positivas les da un aspecto negativo p.e. prioridad 12 -> `nice -12 programa`)
- prioridad detrás de -n (p.e. `nice -n 12 programa`)
- prioridad tras -adjustment= (p.e. `nice -adjustment=12 programa`)

renice

`renice` utiliza los parámetros de la misma forma que nice

- **Consideraciones**
- Cuando se inicia un programa con `nice` sin argumentos este comienza con una prioridad de 10.
- Tanto `nice` como `renice` nos permiten cambiar la prioridad de programas o procesos mediante sin interferir en la ejecución del programa o proceso.
- Si queremos cambiar la prioridad a un proceso, deberemos utilizar el pid de dicho proceso (con man podéis encontrar su sintaxis).
- Podemos cambiar la prioridad de varios procesos a la vez p.e. `renice` prioridad pids -u usuarios
- Podemos utilizar y combinar cambios de prioridad para los procesos independientes con su pid, usuarios y grupos.
- Solo root puede utilizarlos para da incrementar la prioridad.
- Cualquier usuario puede utilizarlos para decrementar la prioridad a los procesos sobre los que tenga permiso

Destruir procesos - kill y killall

Antes hemos visto `nice` y `renice` que nos permiten reducir la prioridad de uno o varios procesos, pero en ocasiones lo que se nos presenta es un proceso que no debería estar ejecutándose, o un programa deje de responder totalmente provocando la aparición de procesos denominados `zombie` (muerto viviente).

Con estos procesos lo que normalmente querremos hacer es destruirlos para ello podemos utilizar `kill`

kill

A `kill` hay que pasarle una señal de finalización, con la que le diremos de que manera queremos que destruya el proceso, podemos pasar tanto el numero como la constante que lo representa, las señales mas habituales son :

Nº. señal	Constante	Descripción
1	SIGHUP	Finaliza los programas interactivos, y hace que muchos demonios vuelvan a leer su fichero de configuración
9	SIGKILL	Finaliza inmediatamente, sin dejar realizar las tareas de finalización
15	SIGTERM	Finaliza el proceso permitiendo realizar las tareas de finalización

Consideraciones

- Debemos tener en cuenta que algunas consolas tienen su propio comando `kill` por lo que seria algo a tener en cuenta
- Para una lista completa así como su sintaxis utiliza `man kill` de esta forma obtendríamos la ayuda

para el `kill` de la consola que estemos utilizando
- Para obtener la lista del comando externo y su sintaxis `man /bin/kill`
- su sintaxis es `kill -s señal pid`
- El valor por defecto es SIGTERM (15)
- Solo podremos destruir aquellos procesos sobre los que tengamos permisos, **esto implica que root debe ser muy cuidadoso, como siempre**.
- El kernel pasa una señal SIGHUP a aquellos procesos que se iniciaron en una sesión al terminar esta, si lo necesitamos, podemos evitar que el proceso termine cuando cerremos sesión, o en situaciones similares, lanzando el programa con el comando `nohup programa opciones`

killall

Nos permite eliminar un proceso en base a su nombre en lugar de su pid, por lo que podemos eliminar todas las instancias de un proceso. Con el parámetro `-i` conseguiremos que se nos pregunte de forma interactiva si queremos eliminar una instancia concreta del proceso o si por el contrario deseamos que continué su ejecución.

En algunos sistemas destruye todos los procesos del usuario que lo ejecuta.

Si lo ejecuta root debe tener cuidado ya que podría eliminar todos las instancias de un proceso aunque no sean suyos, aunque en ocasiones sea exactamente lo que desea.

Es muy recomendable estudiar bien la documentación de estos comandos en cada sistema en particular para familiarizarse y comprender completamente que hacen exactamente en nuestro sistema

Nociones básicas sobre configuración del hardware - Presentación

Este es un tema complejo por lo que debemos tener mucho cuidado ya que si cometemos errores o realizamos tareas de forma inadecuada podremos corromper nuestro sistema y dejarlo inutilizable, por lo que os recomiendo realizar las pruebas (si las hacéis) en un equipo de prueba o en un sistema instalado en una máquina virtual.

Primero unos conocimientos muy básicos sobre algunos conceptos, medios, dispositivos, ... fundamentales, de forma muy superficial ya que serán necesarios en otras secciones para entender que se esta haciendo

BIOS - EFI - OpenFirmware

BIOS (Básic Input Output System) Sistema Básico de Entrada Salida

Normalmente se encuentra en un chip alojado en la placa base de nuestra máquina, este chip empezó siendo una ROM (Read Only Mobile) y evolucionó hasta EEPROM, pasando por las PROM y las EPROM, cada una de ellas añade posibilidades de actualización, configuración, ...

El BIOS como su propio nombre indica es el sistema que se encarga de la gestión de entrada/salida básicas, actúa de interfase entre el S.O. y el hardware. Suele tener un interfase de usuario para poder configurar algunos valores, como activar o desactivar algunas controladoras de dispositivos, controladoras de sonido, puertos, velocidades de reloj, orden de arranque, etc.

Con el tiempo han ido incorporando mas funcionalidades, ya que se han ido añadiendo dispositivos, buses, ... nuevos y mejores.

EFI (Extensible Firmware Interface) Firmware Interfase Extensible, una alternativa mas moderna y potente que la BIOS

OpenFirmware, Alternativa utilizada por Apple, mas potente que BIOS

Unas características interesantes a configurar para servidores son, el que el servidor arranque sin teclado, y que ante un corte de suministro eléctrico la máquina se inicie sola al recuperar el suministro eléctrico (además de otras que os dejo por investigar tanto para servidores como para terminales, sobremesa, ...). Con estas opciones conseguimos un servidor sin teclado evitamos que alguien pulse teclas por error y también entorpecemos la posibilidad de acceso inadecuado, con el arranque automático (si el sistema esta bien configurado y protegido), evitamos tener que ir físicamente a la ubicación del servidor simplemente para apretar un botón.

Para acceder a la interfase de configuración de BIOS ver el manual de nuestro equipo.

Una vez linux se ha iniciado no utiliza el BIOS para las operaciones de E/S. (solamente la utiliza para recopilar información sobre los dispositivos)

IRQ Interruption Request (Petición de Interrupción)

Las IRQs es el mecanismo por el cual los procesos le solicitan a la CPU (Central Process Unit o Unidad Central de Proceso) que "interrumpa" su actividad y ser atiendo.

Las IRQs tienen una jerarquía de prioridades y estan asignadas a cierto tipo de procesos ,que a su vez estan asociados a ciertos dispositivos(reloj tiempo real, teclado, ...).

Bajo ciertas circunstancias algunas pueden ser reasignadas.

En internet es fácil encontrar una lista de ellas.

Si queremos ver que interrupciones esta utilizando nuestro sistema y para que, podemos acceder y ver el fichero `/proc/interrupts`

como vimos en otros apartados podemos hacerlo con `cat`

```
cat /proc/interrupts
```

Ya que linux no registra las IRQs sin usar solo veremos aquellas que estan en uso. Así si utilizamos un hardware que estaba en desuso o desconectado, (y lo conectamos) y volvemos a ejecutar el comando anterior veremos la nueva información

Nos mostrará información en columnas con la interrupción el uso en cada procesador y que driver las usa, una búsqueda en internet nos ayudara a comprender que hace cada driver

Tened mucho cuidado ya que algunos dispositivos no pueden ser conectados en caliente (con la máquina encendida), podrías dañar la máquina, el dispositivo o ambos

/proc es un sistema virtual de archivos no un sistema de archivos físicos. Linux utiliza este sistema para guardar distintos tipos de información del sistema

Direcciones de DMA (Direccionamiento directo de memoria)

Es un método de comunicación con los puertos de E/S. Permite que los dispositivos se comuniquen directamente con la memoria, sin la intervención de la CPU. Con esto se quita carga a la CPU y seguramente se aumente el rendimiento del sistema. Cada canal DMA puede ser utilizado por un único dispositivo.

```
cat /proc/dma
```

nos mostrara los canales DMA en uso.

Direcciones de E/S (Puertos E/S)

Son direcciones de memoria reservadas para la comunicación entre dispositivos físicos hardware y la CPU. Podemos ver que direcciones (puertos) estan en uso examinando el fichero /proc/ioports

```
cat /proc/ioports
```

Nos mostrará información en columnas con las direcciones de memoria utilizadas y que dispositivo la esta utilizando.

Una vez mas internet nos puede ayudar a interpretar los resultados ya que hay algunos dispositivos con nombres poco descriptivos

Tened mucho cuidado ya que algunos dispositivos no pueden ser conectados en caliente (con la máquina encendida), podrías dañar la máquina, el dispositivo o ambos

/proc es un sistema virtual de archivos no un sistema de archivos físicos. Linux utiliza este sistema para guardar distintos tipos de información del sistema

Dispositivos de Arranque - Secuencia de Arranque

Mediante la BIOS podemos decidir, al menos en la mayoría de los casos, la secuencia de arranque, lo que viene a querer decir que creamos una lista de dispositivos con los que probar a iniciar el sistema, como por ejemplo:

1. disquetera
2. cd-rom
3. disco duro 1
4. disco duro 2
5. ...

Ahora bien aunque esta es una secuencia habitual, no es una secuencia recomendable, ya que con esto facilitamos el posible acceso al sistema mediante cualquier medio extraible al sistema. Cualquiera con un disco de arranque bien sea por disquetera o por cd-rom podría reiniciar el sistema para tomar el control del mismo.

Esta configuración es adecuada mientras realizamos la instalación del sistema, o tengamos que realizar operaciones de mantenimiento que así lo requieran. Una vez terminadas estas operaciones lo mejor es modificar esta lista para que solamente pueda arrancar desde el dispositivo (normalmente un disco duro) que hayamos destinado a tal uso.

No voy a entrar en temas de geometrías (CHS, LBA) traducciones CHS por parte de la BIOS, etc. ya que este tema solo daremos unas nociones básicas.

En los sistemas mas modernos podremos configurarlos para que se pueda iniciar desde un pen-drive

Dispositivos de Conexión en Frio/Caliente

Dispositivos de conexión en frío / caliente ¿Que significa esto?

Los dispositivos de conexión en frío son aquellos que solamente se deben conectar o desconectar con la máquina apagada y desconectada. Esto es así, porque su conexión/desconexión con la máquina encendida puede producir una avería tanto en el dispositivo como en la máquina como por ej. dispositivos serie tipo RS232, paralelo,... y por supuesto todos los que van en el interior de la caja como discos duros, microprocesadores, RAM, ...
En este ultimo caso hay que puntualizar, que sobre todo en entornos de servidores profesionales, existen placas base que permiten la conexión de dispositivos en caliente discos duros, micros, memoria, ... y esto es así para facilitar la actualización/ampliación de los equipos sin necesidad de apagar el sistema, ya que en estos entornos se debe evitar al máximo el apagado de la máquina puesto que esto supone dejar a la organización sin sistema informático.

Por el contrario los dispositivos de conexión en caliente son aquellos que se pueden conectar con la máquina encendida, usb, ieee-1394, ethernet, ...

Ficheros y programas a tener en cuenta:

- sysfs montado en /sys
- /dev
- HAL esto es un "demonio"
- D_Bus otro demonio
- udev sistema de archivos virtual montado en /dev
- ...

¿Qué es un demonio?

Es la denominación que utiliza linux para referirse a un programa/proceso que se esta ejecutando siempre y en las "tinieblas" (no profundizo mas)

¿Qué es montar un dispositivo?

Linux trata los dispositivos como ficheros, por lo que cuando se conecta uno se "monta" en una parte de nuestro sistema de archivos para poder acceder a él (tampoco profundizo mas)

En los sistemas modernos los dispositivos se montan y desmontan automáticamente, al menos en sistemas de hogar, en los sistemas profesionales bien administrados, incluso a algunos usuarios les resultara imposible poder utilizar dispositivos extraíbles, puesto que el administrador del sistema no les permitirá esta opción. Así se evita que se pueda extraer información, de cualquier tipo, del sistema por ese usuario/grupo de usuarios. Esto es así porque por la naturaleza del roll del usuario en cuestión no necesita poder introducir/sacar información del sistema bajo ningún concepto.

Configurar tarjetas PCI

Las tarjetas PCI se conectan a un bus del mismo nombre, que es el bus de expansión estándar para la mayoría de dispositivos internos. Al contrario que los dispositivos antiguos en los que su configuración se debía hacer mediante jumpers, microswitch, ... , actualmente y gracias a los dispositivos PCI los cuales fueron concebidos para una configuración automática o plug-and-play estas configuraciones (IRQ,DMA,Puertos) se realizan mediante software.

Resumiendo estas tarjetas de expansión son capaces de analizar nuestro sistema y utilizar aquellas configuraciones que están libres, o que no deberían causar ningún conflicto, aunque nos podemos encontrar con algunas configuraciones, no correctas para nuestro sistema.

- El kernel o núcleo de linux nos permite indicarle en que forma se detectarán los dispositivos PCI, Bus Options>PCI Acces Mode (esto lo podremos hacer en la compilación del kernel). Esta opción nos permite las siguientes opciones para detectar los dispositivos:

1. **BIOS** Emplear la BIOS.
2. **MMConfig** Utilizar un protocolo del mismo nombre.
3. **Direct** método propio de linux.
4. **Any** Cualquiera en el siguiente orden MMConfig, Direct, BIOS

Si nuestros dispositivos no se detectan adecuadamente podemos experimentar con esta opción del kernel.

- La BIOS suele tener opciones de PCI que permiten modificar el modo en que se asignan los recursos, esta opción solamente será de utilidad si tememos nuestro kernel con la opción de BIOS.

- Algunos drivers de linux admiten opciones, tendremos que consultar su documentación y luego pasárselo al kernel empleando un cargador de arranque, o como opciones del módulo de kernel correspondiente.
- `setpci` utilidad para consultar y ajustar directamente las configuraciones de los dispositivos PCI, esta herramienta es más útil tanto mas conozcamos de nuestro hardware.

Ademas de las opciones de configuración, podemos utilizar el comando `lspci`, que dispone de varias opciones, para comprobar como están configurados los distintos dispositivos.

Información sobre los módulos del kernel

Los dispositivos hardware los administran/manejan los drivers del kernel, la mayoría de estos vienen en forma de módulos del kernel. Se tratan de ficheros independientes, que como norma suelen estar guardados en el directorio `/lib/modules` . Normalmente Linux cargará los módulos necesarios para nuestro hardware, aunque en caso de ser necesario podremos cargar y descargar módulos manualmente según nuestras necesidades (p.e. un dispositivo no funciona correctamente, existen funcionalidades del dispositivo que no podemos utilizar o simplemente no funciona en absoluto).

Podemos saber qué módulos están cargados en el sistema utilizando `lsmod`. No recibe argumentos y genera una salida tabulada, realizad una prueba para ver la información que nos facilita (tal y como indica la información facilitada por man sobre este comando nos presenta información existente en `/proc/modules` formateada).

Francisco Javier López Torrijos
Analista Sistemas Informáticos de Gestión
Diseño y Desarrollo Web

Enlaces
[1] http://www.rpm.org
[2] https://fedoraproject.org/wiki/FedoraProject:Translating/es
[3] http://wiki.debian.org/es/Apt