Configuración y administración de un sistema GNU/Linux

Margarita Manterola y Maximiliano Curia

Actualizado Marzo 2003

Índice

1. Introducción

2. Configuración del Sistema

En esta sección vamos a ver, sin adentranos mucho en detalles ni conceptos, como configurar nuestro equipo, para distintas situaciones típicas. Si bien normalmente las configuraciones que ofrecemos son independientes de la distribución de GNU/Linux que usemos, cuando tengamos que dar detalles los daremos para Debian GNU/Linux.

2.1. Obtener información

Siempre que queramos configurar un dispositivo será importante tener información sobre sus características. Para esto, puede resultar de utilidad, consultar los manuales, la caja o mirar la plaqueta del dispositivo. GNU/Linux tiene, además, ciertas herramientas para ayudarnos a configurar nuestro equipo.

lspci nos muestra información sobre las placas PCI que tengamos instaladas en nuestro sistema. Es, en realidad, un resumne de la información mas útil que se encuentra en /proc/pci.

pnpdump nos muestra información (mucha) sobre las placas isapnp que tengamos (muchas veces, ninguna). Además, la salida de este comando nos sirve para tener un archivo de configuracion para **isapnp**.

discover puede darnos información acerca del hardware que tenemos, según el tipo de harware que estemos buscando.

Ejemplo **discover sound** nos dice qué placa de sonido tenemos y si ejecutamos **discover –module sound** nos dice qué módulo de kernel deberemos utilizar para configurar esa placa.

También podemos preguntarle sobre: bridge, cdrom, ide, scsi, usb, ethernet, modem, y video.

mdetect nos permite detectar que tipo de mouse tenemos.

read-edid , para configurar el video.

kudzu , para todo tipo de hardware.

2.2. Configuración de mouse en consola

El mouse es un dispositivo de entrada muy útil, que nos puede ayudar tanto en entornos gráficos como en consola de texto. Para utilizar el mouse dentro de la consola de texto utilizamos un servicio llamado **gpm** (General Purpose Mouse Interface).

Además de disponibilizar el uso del mouse para muchos programas también nos dá algunas ventajas extras como la de seleccionar una fragmento de texto y pegarla con el botón del medio. El programa **gpm** no tiene definido un archivo de configuración, sino que recibe los valores por parámetros.

Por ejemplo: `/usr/sbin/gpm -m /dev/psaux -t imps2`, le dice que use el dispositivo **/dev/psaux** como un mouse *IntelliMouse* PS/2.

Como tantos otros programas dentro del entorno Unix, **gpm** es un programa que realiza una sola tarea, la de manejar el mouse, y lo hace bien. Mientras que el servidor **X** de modo gráfico hace una gran cantidad de cosas. Si elegimos que sea **gpm** quien maneje el mouse, podemos aliviar parte del trabajo del X. Para eso existe la opción -R de gpm (repeat) que puede enviar los comandos que detectó el gpm con o sin formato a un dispositivo creado por el gpm (**/dev/gpmdata**). Por ejemplo: `/usr/sbin/gpm -m /dev/psaux -t imps2 -Rms3`.

Para configurar este servicio, dentro de la distribución Debian GNU/Linux se utiliza el script **gpmconfig**. En otras distribuciones se utilizan otros programas para configurar, o se edita un archivo de configuración. Los valores a utilizar serán muy similares, ya que en definitiva son parámetros que se le indican al **gpm**.

El **gpmconfig** es un script interactivo que en primer lugar muestra la configuración actual del mouse (si es que existe), y luego permite modificarla según sea necesario. Los parámetros que deben ser configurados son:

Dispositivo (device) es el dispositivo al que está conectado el mouse. Las posibilidades son las siguientes.

- **/dev/psaux**, si se trata de un mouse PS/2.
- **/dev/ttyS0**, si está conectado al COM1.
- **/dev/ttyS1**, si está conectado al COM2.
- **/dev/ttyS2**, si está conectado al COM3.
- **/dev/ttyS3**, si está conectado al COM4.
- **/dev/usb/mouse**, si es un mouse USB.
- **/dev/mouse**, que normalmente es un symlink al dispositivo correspondiente al mouse.

Tipo (type) es el tipo de mouse que se quiere configurar (con 2 o 3 botones, con o sin ruedita, etc). En este caso hay muchas posibilidades, se incluyen aquí las más comunes.

- **msc**, para la mayoría de los mouse serie de 3 botones.
- **ms**, si se trata de un mouse serie, de tipo *Microsoft*, con 2 o 3 botones (si tiene sólo dos, el tercero es *simulado* cuando se apretan los dos a la vez).
- **ms3**, si se trata de un mouse serie con ruedita y tres botones.
- **ps2**, si se trata de un mouse PS/2 genérico.
- **imps2**, si se trata de un mouse PS/2, con ruedita y dos o tres botones (si tiene dos, el tercero es emulado).
- **netmouse**, si se trata de un mouse *Genius Netmouse*, que tiene dos botones de arriba/abajo en lugar de ruedita.

Aceleración (responsiveness) permite que el mouse se mueva a mayor velocidad a través de la pantalla. Se trata de un valor númerico, cuyo valor predeterminado es 10. Un cursor veloz se obtiene con 20, utilizando 30 ya es demasiado rápido (todo esto depende del mouse).

Protocolo de Repetición (repeat protocol) es la forma en que el **gpm** va a repetir la entrada del mouse al modo gráfico, según se explicó anteriormente.

Para que esto se haga efectivo, será necesario indicarle al servidor **X** que la entrada de mouse la lea de **/dev/gpmdata**, en lugar de leerla directamente desde el mouse. Y el tipo de mouse que se le especifique al servidor deberá coincidir con el protocolo indicado.

Si se especifica la opción **none**, no se realizará la repetición. Esta es la opción a seleccionar cuando se quiere que el modo gráfico maneje la entrada de mouse. Si, en cambio, se especifica la opción **ms3**, la repetición se realizará en el protocolo *IntelliMouse* y esto mismo habrá que seleccionar en la configuración del servidor.

Opciones Adicionales (aditional arguments) es posible configurar una cantidad de opciones adicionales a las mencionadas anteriormente. Para saber qué opciones pueden utilizarse en esta línea, se puede consultar el manual de **gpm**.

2.3. Configuración de teclado

La forma de configurar el teclado varía de distribución en distribución. Hay dos maneras de configurarlo: directamente desde el kernel o desde el sistema. Algunas distribuciones utilizan una forma o la otra, y algunas -como Debian- permiten elegir de qué forma se lo configurará.

Dentro de Debian la forma de configurar el teclado es ejecutar el comando **dpkg-reconfigure console-common**. La primera pantalla explica las opciones que se van a mostrar en la segunda. En la segunda pantalla se permite seleccionar la forma de configuración de teclado (kernel o no).

Seleccionando la opción **Select keymap from arch list**, debemos luego poner la disposición general de las teclas (**querty** es la disposición común, de la mayoría de los teclados en inglés y español). Y en la siguiente pantalla se selecciona la disposición regional. El teclado **latinoamericano** es el teclado que tiene la @ en la misma tecla que la Q, y el **español** el que tiene la @ en la misma tecla que el 2 y las ".

2.4. Módulos

En GNU/Linux, gran parte del hardware que utilicemos tiene que estar soportado por el kernel Linux, antes de que lo usemos.

Por ejemplo, normalmente Linux ya tiene compilado el soporte de rígidos IDE, de manera que podemos utilizar el disco rígido ni bien se inicia el sistema. Esto mismo sucede con los dispositivos más comunes, como los puertos serie y paralelo.

Sin embargo, es normal que no este compilado en el kernel el soporte para todo el resto del hardware, sino que el soporte esté disponible en forma de módulos. En la práctica los módulos funcionan como los *drivers* en otros sistemas operativos, cargar el módulo correcto nos permite utilizar nuestro hardware.

Para manejar los modulos usamos:

lsmod Lista los módulos que ya están cargados.

modprobe -l Muestra todos los módulos disponibles para la versión del kernel que estamos utilizando. (El listado es largo, se puede utilizar: **modprobe -l — less**, o bien **modprobe -l — grep net**, **modprobe -l — grep agp**, según el módulo que se esté buscando).

modinfo módulo Muestra información sobre el modulo, como por ejemplo, qué opciones recibe.

modprobe módulo opciones Carga el módulo y le pasa las opciones.

insmod ruta/modulo.o Otra forma de cargar el módulo, en este caso recibe como parámetro la ruta de acceso al módulo.

rmmod Quita el módulo de memoria. El módulo no puede estar siendo utilizado si se lo quiere sacar de la memoria.

modconf (Específico de Debian) Es una aplicación un poco más amigable, que permite buscar en el listado de módulos, instalar el módulo que necesario y configurarlo para que se cargue automáticamente al reiniciar el sistema.

Por otro lado, si queremos configurar los módulos para que se carguen cuando se inicia el sistema, deberemos editar el archivo **/etc/modules**.

Y también el archivo **/etc/modules.conf**, que permite configurar algunas opciones que el sistema le va a pasar a los módulos cuando se cargan, así como formas para que cargue determinado módulo al querer usar determinado recurso.

Las sintaxis de este archivo es:

options módulo opciones Para especificar las opciones para el módulo.

alias recurso módulo Para especificar qué módulo cargar al usar determinado recurso.

En Debian GNU/Linux el archivo /etc/modules.conf es mantenido por una aplicación llamada **modutils**, que permite tener varios archivos de configuración sencillos, separados por tareas o recursos. Estos archivos deben ser colocados dentro del directorio **/etc/modutils/** . Y para recopilar los archivos en /etc/modules.conf se utiliza el comando **update-modules**.

2.5. Placas de Red - Ethernet

La mayoría de las placas de red PCI se configuran sin mayores dolores de cabeza, mientras que para configurar las ISA puede llevar más tiempo encontrar la combinación de direccion/irq apropiada.

2.5.1. Configuración del módulo

En el caso de las placas PCI con la ayuda de **discover** se puede obtener el nombre del módulo que debe ser cargado para habilitar la placa. Para la mayoría de las placas ISA se deberá utilizar el módulo **ne**.

Algunos ejemplos, de configuración de placas de red.

modprobe ne io=0x300 irq=10 Le indica al sistema que busque y use una placa de red ISA compatible con ne2000 el la dirección (io) 300, irq 10. La dirección de la placa debe escribirse en hexadecimal, a eso se debe el 0x al comienzo del valor.

modprobe ne2k-pci Le indica al sistema que habilite una placa PCI, la mayoría de las placas de 10mb/s se configuran con este módulo.

2.5.2. IP

Una vez cargado el módulo tendremos disponible el acceso al dispositivo, pero todavía no habremos configurado el acceso a la red.

Los nombres de los dispositivos que representan a las placas de red son eth0 para la primera placa de red, eth1 para la segunda, etc.

Si en nuestra red se utiliza DHCP (obtención dinámica de IPs), deberemos ejecutar un cliente de DHCP como **pump**, **dhcpcd** o **dhclient**, para obtenerla. Estos comandos reciben como parámetro el nombre del dispositivo a configurar.

En el caso de tener que configurar una dirección de IP manualmente, el comando ifconfig nos puede resultar de utilidad. Si se ejecuta sin parámetros muestra el estado de los dispositivos de red (además del dispositivo especial **lo**), pero también nos sirve para configurar una placa de red.

Por ejemplo: **ifconfig eth0 192.168.200.32 netmask 255.255.255.0 up**, configura la dirección 192.168.200.32 para nuestra placa de red eth0.

2.5.3. Rutas y DNS

Normalmente también deberemos configurar un gateway, esto es, la dirección de la máquina que nos permite acceder a otras redes. El comando **route** será el que deberemos utilizar. Si lo ejecutamos sin parámetros nos muestra información, al igual que **ifconfig**, y con los parámetros adecuados podemos configurar la ruta deseada.

Por ejemplo: **route add default gw 192.168.200.1**, agrega una ruta default (normalmente, el acceso al resto del mundo) a través de 192.168.200.1, que deberá ser nuestro gateway.

También debemos configurar un DNS (Domain Name Server), es decir, la IP de una máquina que pueda transformar nombres de dominios en direcciones IP. Para eso debemos editar el archivo **/etc/resolv.conf**, la sintaxis es simple. Por ejemplo:

```
domain local
nameserver 200.42.0.108
nameserver 200.42.0.109
```

La primera línea configura el dominio en el que se encuentra la estación. El dominio en el que esté dependerá de qué tipo de red se trate. Las otras dos líneas de nameserver configurar los DNS que vamos a utilizar.

2.5.4. Proxy

Si estamos en una red en la que debemos utilizar un Proxy para poder acceder a Internet, deberemos configurar el proxy desde cada programa que queramos utilizar.

Existe una variable de entorno, **http_proxy**, que es utilizada por los programas de consola, como por ejemplo el navegador **lynx**. En esta variable deberemos poner la ruta al proxy, incluyendo el usuario y el password (si es que el proxy lo requiere).

Por ejemplo: **http_proxy=http://usuario:password@host.dominio:8080**. Donde, host.dominio es el nombre de la máquina que funciona como proxy en nuestra red, y 8080 es el puerto donde se encuentra el proxy. Otros puertos comunes donde podemos encontrar un proxy son: 3128 y 80.

Los navegadores gráficos, como el mozilla, tendrán una sección dentro de las preferencias donde deberá introducirse la dirección del proxy y el puerto a utilizar.

Para otros programas, deberemos utilizar el manual de la aplicación para encontrar de qué manera se configura el proxy. Por ejemplo, en el caso de la herramienta apt de Debian, el proxy se configura en el archivo **/etc/apt/apt.conf** y la sintaxis es:

```
Acquire::http::Proxy "http://usuario:password@host.dominio:8080";
```

2.5.5. Configuración en Debian

Hasta aquí, las configuraciones que hicimos fueron cargadas a mano. Sin embargo, lo deseable es realizar todos estos pasos una sola vez y que luego se carguen automáticamente al iniciar el sistema. Esta etapa será distinta según la distribución que estemos utilizando.

En Debian GNU/Linux, utilizaremos la herramienta **modutils**, que mencionamos antes, para que se carge el módulo de la placa automáticamente cuando vamos a usar la placa. Para eso creamos un archivo cualquiera (por ejemplo eth0) en **/etc/modutils** con una o dos líneas:

```
alias eth0 módulo
options módulo opciones
```

Deberemos utilizar la línea **options** solamente en el caso de que nuestra placa requiera opciones adicionales al utilizar **modprobe** (por ejemplo, si es una placa ISA).

Una vez editado el archivo, deberemos ejecutar **update-modules**, de forma que estos datos se guarden en **/etc/modules.conf**. De esta forma, cuando tratemos de usar eth0 cargará automaticamente ese módulo dejando disponible eth0.

A continuación, editaremos el archivo **/etc/network/interfaces**.

Este archivo tiene una línea **auto** que dice que interfaces configurar cuando inicia el sistema. Varias líneas **iface** una por cada dispositivo de red, además de una especial (lo), que es importante tener.

En las líneas **iface** podemos configurar si la interfaz se configura manualmente (static) o dinámicamente (dhcp). Si ponemos static deberemos especificar dirección (address), máscara de red (netmask) y gateway. Por ejemplo:

```
auto lo eth0

iface lo inet loopback

iface eth0 inet static
          address 192.168.200.32
          netmask 255.255.255.0
          gateway 192.168.200.1

iface eth1 inet dhcp
```

Una vez editado el archivo, utilizaremos el comando **ifup intefaz** para levantar la configuración de una determinada interfaz, e **ifdown interfaz** para bajarla.

Es importante recordar que si configuramos nuestra red static deberemos editar el archivo **/etc/resolv.conf** para ingresar los DNS, como se explicó anteriormente.

2.6. Sonido

Configurar la placa de sonido, suele ser bastante sencillo. Será necesario, al igual que con la placa de red, cargar el módulo correcto, con las opciones necesarias.

Algunos ejemplos de módulos a utilizar:

modprobe trident es el módulo para las placas de sonido SiS701x y para las placas de sonido Trident.

modprobe cmpci es el módulo que se utiliza para muchas placas C Media.

modprobe sb io=0x220 irq=5 dma=1 dma16=5 será la línea que se deberá utilizar para configurar una Sound Blaster 16 ISA, con algunas de las opciones comunes.

Una vez que hayamos probado que la placa está funcionando correctamente, tendremos que agregar el módulo a **/etc/modules** para que pueda utilizarse cada vez que se inicia la computadora

Una posibilidad para probar si la placa de sonido está funcionando o no, es el comando de consola **play**, que viene dentro del paquete **sox**. Podremos usar **play /usr/share/sounds/*.wav** para verificar el funcionamiento del sonido.

La placa de sonido en Debian GNU/Linux sólo la pueden usar aquellos usuarios que pertenezcan al grupo *audio*. Para agregar a un usuario a ese grupo, utilizamos: **adduser usuario audio**.

2.7. Video

La forma mas común de usar la placa de video en modo gráfico es usando el Xfree86, y para poder utilizarlo no hace falta cargar módulos de kernel. Es necesario, sin embargo, configurar en XFree86 una serie de parámetros que serán utilizados por la placa de video.

Veremos cómo configurar un Xfree86 4.x.x y luego veremos las diferencias para versiones anteriores.

2.7.1. XFree86 4.x.x

El archivo de configuración que vamos a editar es **/etc/X11/XF86Config-4**. Dado que es un archivo bastante largo y con muchas posibilidades distintas, está divido en secciones que permiten configurar el entorno, el mouse, el teclado, el monitor, la placa de video, la resolución, etc. Además tiene una seccion que indica cómo combinar las anteriores (layout).

Si se arman varias resoluciones distintas, una vez que estemos utilizando el X podremos cambiar entre ellas con CrtlAlt+ y CrtlAlt-. Si queremos cerrar el X por la fuerza, podemos utilizar Ctrl-Alt-Backspace.

Veremos ahora las secciones que son relevantes a la configuración del video.

```
Section "Device"
    Identifier  "Primary Card"
    BoardName   "NVIDIA Riva TNT2 (generic)"
    Driver      "nv"
    VideoRam    32768
EndSection
```

En la sección **Device** se configura la placa de video. La línea más significativa es **Driver**, es la que indica qué controlador usar para esta placa. Con **Identifier** le damos un nombre, con el cual haremos referencia a esta placa más adelante. La linea **BoardName** es solo descriptiva. Por último, la línea **VideoRam** es opcional, aunque puede ser necesario para ciertas placas no muy bien soportadas.

Para obtener una lista de placas soportadas podemos usar **discover**, buscarlo en la web o revisar en los archivos que nos instaló el paquete. Generalmente hay un archivo que contiene la lista de placas compatibles con cada driver.

En Debian GNU/Linux tenemos páginas de **man** para cada uno de los controladores de placas de video. Además, podemos utilizar **dpkg -L xserver-xfree86** para ver una lista de los archivos que son de ese paquete.

```
Section "Monitor"
    Identifier  "My Monitor"
    HorizSync   30 - 55
    VertRefresh 50-120
EndSection
```

Una vez especificada la placa de video, debemos asegurarnos que la sección **Monitor** esté bien configurada. Para eso debemos ponerle el rango de frecuencias horizontales y verticales. Puede ser necesario que consultemos el manual del monitor para obtener estos valores.

Por otro lado, muchos de los monitores modernos pueden informarle a la placa de video estos datos sin nuestra intervención. En ese caso, es necesario agregar `Load "ddc"`, a la sección **Module**.

Además, si en lugar de editar el archivo de configuración directamente, utilizamos algún programa para configurar el XFree86, probablemente contaremos con una base de datos con los valores necesarios para gran cantidad de monitores.

```
Section "Screen"
    Identifier  "Screen 1"
    Device      "Primary Card"
    Monitor     "My Monitor"
    DefaultDepth 16
    Subsection "Display"
        Depth       8
        Modes       "640x480" "800x600" "1024x768"
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth       16
        Modes       "1024x768" "640x480" "800x600"
        ViewPort    0 0
    EndSubsection
```

```
    Subsection "Display"
        Depth       24
        Modes       "640x480"
        ViewPort    0 0
    EndSubsection
EndSection
```

La sección **Screen** relaciona la configuración de la placa de video con la del monitor, configura la cantidad de colores estándar (normalmente no se puede cambiar una vez iniciado el X) y las resoluciones a usar.

Esta configuración está pensada para que sea sencillo configurar varios monitores y placas de video, tener múltiples monitores por usuario o manejar distintas consolas gráficas.

2.7.2. Xfree86 3.x.x

Si en lugar de utilizar la versión 4 de XFree86, utilizamos una versión 3.x.x, debemos tener en cuenta que el comando que se ejecutará (xserver) será distinto según la placa de video que utilicemos.

En Debian GNU/Linux (y también en otras distribuciones) existen varios paquetes distintos, llamados xserver-* (xserver-svga, xserver-ati, etc) que son los comandos correspondientes a cada placa. Mientras que xserver-xfree86 es el xserver de Xfree86 4.x.x.

Además, cuando utilizamos la versión 3 de XFree86, el archivo de configuración será **/etc/X11/XF86Config**.

La sintaxis de este archivo es levemente distinta. Para la misma configuración que vimos anteriormente tendremos:

```
Section "Monitor"
   Identifier      "Primary Monitor"
   HorizSync       31.5-48.5
   VertRefresh     55-90
EndSection
```

Es decir, en la configuración del monitor casi no hay diferencias con respecto a la configuración anterior. Las diferencias principales estarán en las otras dos secciones.

```
Section "Device"
   Identifier      "Primary Card"
   VendorName      "Unknown"
   BoardName       "NVIDIA Riva TNT2 (generic)"
```

```
EndSection

Section "Screen"
   Driver          "Accel"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 16
   SubSection "Display"
      Depth        16
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        24
      Modes        "1024x768"
   EndSubSection
EndSection

Section "Screen"
   Driver          "SVGA"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   DefaultColorDepth 16
   SubSection "Display"
      Depth        16
      Modes        "1024x768"
   EndSubSection
   SubSection "Display"
      Depth        24
      Modes        "1024x768"
   EndSubSection
EndSection

Section "Screen"
   Driver          "VGA16"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   SubSection "Display"
      Depth        4
      Modes        "1024x768"
   EndSubSection
EndSection
```

```
Section "Screen"
   Driver          "VGA2"
   Device          "Primary Card"
   Monitor         "Primary Monitor"
   SubSection "Display"
      Depth        1
      Modes        "1024x768"
   EndSubSection
EndSection
```

Como vemos en este último tramo del archivo, en la sección **Device** no se especifica el Driver a utilizar, sino que se especifica en la sección **Screen**. Los drivers con los que contemos para configurar nuestra placa de video, dependerán del xserver que estemos utilizando. Por ejemplo, los que están aquí listados (Accel, SVGA, VGA16 y VGA2) son los drivers que nos permite utilizar xserver-svga.

2.7.3. Aplicaciones de Configuración

El XFree86 nos provee de una herramienta gráfica para poder escribir en el archivo de configuración, llamada **xf86config**.

En Debian GNU/Linux podemos utilizar una herramienta del sistema para configurar el XFree86, sin tener que editar el archivo de texto. Accedemos a esta herramienta utilizando **dpkg-reconfigure xserver-xfree86** (o xserver-svga, o xserver-ati, o el xserver que estemos utilizando).

2.7.4. Mouse

Para configurar el soporte de mouse para X, utilizamos una sección dentro de XF86Config o XF86Config-4:

```
Section "InputDevice"
    Identifier  "Mouse1"
    Driver       "mouse"
    Option      "Protocol"    "IntelliMouse"
    Option      "Device"      "/dev/mouse"
    Option      "Buttons"    "3"
    Option      "ZAxisMapping" "4 5"
EndSection
```

Aquí le decimos que utilice el dispositivo **/dev/mouse**, se puede utilizar cualquiera de los dispositivos que se explicaron anteriormente, incluyendo **/dev/gpmdata**. Además, se está utilizando el driver de *IntelliMouse*. La indicación de que los botones son 3 permite que el tercer botón sea emulado apretando los dos botones a la vez, si el mouse no tiene más que dos botones. Por último, se habilita la ruedita (ZAxisMapping 4 5).

Para que esto funciones, deberemos asegurarnos que el dispositivo **/dev/mouse** existe (generalmente es un symlink al dispositivo real).

El soporte de mouse en X es muy amplio, para más información se puede consultar el archivo README.mouse en la distribucion de Xfree86 (viene con los paquetes de xserver también).

3. Instalación de Software

El software libre se distribuye en muchas formas y formatos distintos. Por ejemplo, el conocido navegador de internet Mozilla lo podemos encontrar en formato binario para Windows, MacOs y GNU/Linux; también lo podemos encontrar empaquetado para diversas distribuciones de GNU/Linux como RedHat, Mandrake, Debian, Slackware, etc; así como también podemos obtener el código fuente del navegador y compilarlo en la plataforma que tengamos.

A continuación daremos un repaso general en cuanto a cómo manejar cada uno de estos distintos formatos.

3.1. Formatos de compresión

Uno de los formatos de compresión más usuales en el mundo del software libre, es **.tar.gz** o **.tgz**. Este formato es, en realidad, la combinación de dos formatos.

El formato **.tar** agrupa varios archivos en uno solo, de tal manera que se pueden conservar los datos extendidos de los archivos (permisos, dueño, grupo, fechas). Originalmente, fue pensado para grabar los archivos sin compresión, en una cinta de backup.

El formato **.gz** es el que se encarga de la compresión. El algoritmo de compresión es muy similar al del **.zip**.

Además del **.tar.gz** existe el **.tar.bz2** que se diferencia en que el formato de compresión es el del **bzip2**.

Para descomprimir estos formatos podemos utilizar dos formas alternativas:

- `gunzip -c archivo.tar.gz | tar x`
- `tar xzf archivo.tar.gz`

De la misma manera, para crear un archivo comprimido, podemos elegir de varias formas alternativas:

- `tar cf - comprimir* | gzip -c - > archivo.tar.gz`
- `tar cf archivo.tar comprimir* ; gzip archivo.tar`
- `tar czf archivo.tar.gz comprimir*`

Así vemos como el comando **tar** nos dá la posibilidad de utilizar los comandos **gzip** y **gunzip** directamente desde sus opciones. De la misma manera, podemos utilizar los comandos de **bzip2** y **bunzip2** con la opción **j**.

El conocido formato **.zip**, puede utilizarse con los comandos **zip** y **unzip**. Los archivos generados con este formato serán compatibles con los generados con otras plataformas (Ej: WinZip).

3.2. Desde el principio: utilización de código fuente

Si un determinado programa tiene una licencia de software libre, significa que de alguna manera podremos llegar a obtener el código fuente de ese programa.

Normalmente el código fuente se encuentra comprimido en el formato **.tar.gz**. Una vez que hayamos descomprimido los archivos, deberemos ejecutar la acción que corresponda según el lenguaje en el que esté desarrollada la aplicación.

La mayor parte del software libre está desarrollada en lenguaje C, y pensada para ser compilada con el **gcc** (GNU C Compiler).

El software que sigue los lineamientos de empaquetado de GNU, tendrá un comando **configure** que detectará una gran variedad de datos acerca de nuestro sistema (el procesador, el sistema operativo, el compilador C, las bibliotecas necesarias para compilar, y todos los recursos necesarios para compilar el código fuente en cuestión). En el caso en que falte un determinado recurso (bibliotecas, programas, etc), nos avisará que deberemos obtenerlo.

Una vez que todos los recursos han sido detectados correctamente, será necesario ejecutar el comando **make**, que compila el código fuente, y luego el comando **make install** que instala el programa en nuestro sistema, dejándolo listo para usar.

Para el caso de los programas que no usen **configure** y **make** será necesario leer la documentación que acompañe al código fuente para saber cómo realizar la compilación.

Una de las excepciones comunes a los lineamientos de empaquetado de GNU es el kernel (Linux), que incluye un menú de configuración propio, y los pasos de instalación son distintos. Veremos cómo realizar la compilación y la instalación más adelante en este curso.

3.3. Ganando tiempo: utilización de código binario

Si bien todo programa que sea software libre nos da la posibilidad de que lo compilemos nosotros mismos, esto requiere mucho tiempo, y normalmente no hay ganancia en el rendimiento que tiene la aplicación. De manera que muchas veces podemos elegir utilizar el código binario que ya ha sido compilado por otras personas, para la plataforma que estemos utilizando.

La gran mayoría de los programas, se distribuyen también en forma binaria, compilada por el mismo desarrollador. Y normalmente este código binario se encontrará en formato .tar.gz, al igual que el código fuente. O incluso en archivos ejecutables que pueden instalarse directamente.

Por lo general alcanza con descomprimir el archivo y luego agregar el directorio correspondiente a la variable PATH, o bien ejecutarlo directamente desde el directorio.

Tanto la instalación desde el código fuente como la instalación a partir del código binario permiten que un usuario instale una aplicación en su directorio personal sin tener que pedirle permiso al administrador del sistema.

Cuando la instalación es realizada por el administrador del sistema, es recomendable colocar los programas en la ruta **/usr/local**, o bien **/opt**. De forma que todos los usuarios del sistema puedan acceder a estos programas.

3.4. Manejo de paquetes

Llamamos *paquete* al conjunto formado por el código binario de una aplicación, los scripts necesarios para configurar, instalar y desinstalar esta aplicación, los datos acerca de que otros programas y bibliotecas que son necesarios para su correcto funcionamiento (dependencias) y algunos otros datos adicionales relacionados con la aplicación en cuestión.

Existen varios formatos de paquetes, los de Red Hat y derivados (rpm), los de Debian (deb), los de Slackware (tgz). Para manejar estos paquetes, cada distribución tiene su conjunto de herramientas, que permiten instalarlos, desinstalarlos, actualizarlos, etc.

3.4.1. Paquetes rpm

En el caso de los paquetes rpm, la herramienta para manejarlos se llama también **rpm** y para instalar un paquete debemos primero obtenerlo (por ejemplo, de Internet) y luego ejecutar **rpm -i paquete.rpm**.

Para actualizar un paquete que ya está instalado en el sistema, utilizamos el comando **rpm -iU paquete.rpm** (en el caso en que el paquete no estuviera instalado, el funcionamiento será el mismo que -i).

Para desinstalar un paquete ya instalado, utilizaremos el comando **rpm -e nombre-paquete**.

Varias distribuciones aparte de Red Hat (Mandrake, Suse, Conectiva, etc) utilizan paquetes rpm, sin embargo no es recomendable utilizar en una distribución un paquete pensado para otra distribución -incluso distintas versiones de una misma distribución- pues la resolución de las dependencias suele ser muy distinta y puede dar lugar a problemas variados.

Para obtener paquetes rpm en internet, el sitio más utilizado es: **http://www.rpmfind.net**. En este sitio se pueden buscar el archivo rpm para el paquete y la distribución deseada.

Existe un sistema llamado up2date que nos permite bajar las actualizaciones de los paquetes de Red Hat, esta aplicación utiliza una base centralizada en los servidores de Red Hat, aunque ofrece un número limitado de conexiones anónimas (gratuitas).

Es de esperar que aplicaciones similares existan, o bien se estén desarrollando, para otras distribuciones de paquetes rpm.

3.4.2. Paquetes deb

En el caso de los paquetes deb, contamos con dos herramientas para instalar, desinstalar y configurar los paquetes: **apt** y **dpkg**.

El **dpkg** es un manejador de paquetes muy parecido al **rpm** (tiene opciones para instalar, actualizar, desinstalar, etc). Puede ser utilizado obteniendo los paquetes y luego escribiendo: **dpkg -i paquete.deb** para instalar o actualizar, **dpkg -r nombre-paquete** para desinstalar.

Sin embargo, la herramienta más utilizada (por su sencillez de uso) es **apt**. Con esta aplicación podemos buscar paquetes por su nombre o por su descripción, utilizando el comando **apt-cache search patrón** y luego instalar los paquetes obteniéndolos directamente de Internet o de un cd-rom, cumpliendo con todas las dependencias que sean necesarias, con **apt-get install nombre-paquete**.

Para esto será necesario configurar un archivo /etc/apt/sources.list, donde se encuentran los datos que necesita el programa para saber de dónde obtener los paquetes. Este archivo puede escribirse en forma manual, o a través de la herramienta **apt-setup**. Un ejemplo de algunas líneas del archivo sources.list, serían:

```
deb http://http.us.debian.org/debian sarge main contrib non-free
deb http://non-us.debian.org/debian-non-US sarge/non-US main contrib non-free

deb cdrom:[Debian GNU/Linux 2.2 r0 _Potato_ - Official i386 Binary-1
(20000814)]/ unstable contrib main non-US/contrib non-US/main
```

Una vez que se ha configurado correctamente este archivo, será necesario ejecutar **apt-get update** para tener un listado actualizado de los paquetes disponibles.

En el caso en que se utilizan paquetes de internet, es recomendable actualizar este listado periódicamente.

Con la lista de paquetes actualizada podemos pedirle a sistema que actualice todos los paquetes que tenemos instalados a la última versión con **apt-get upgrade** o **apt-get dist-upgrade** , la diferencia es sutil: si alguno de los paquetes que queremos actualizar necesita de otro que no tenemos instalado el upgrade no lo actualizará, en cambio el dist-upgrade instalará el nuevo paquete y actualizará el que ya estaba instalado.

Recientemente, se ha incorporado una herramienta similiar a apt para Red Hat, llamada apt-rpm.

3.4.3. Paquetes Slackware

Los paquetes Slackware utilizan la extensión **.tgz**, pero están pensados para ser utilizados con las herramientas propias de Slackware para su instalación, actualización y desinstalación.

Las herramientas utilizadas son **pkgtool**, una aplicación que permite seleccionar los paquetes a instalar o desinstalar, desde un menú amigable. Y, por otro lado, **installpkg** para instalar los paquetes, **upgradepkg** para actualizarlos y **removepkg** para desinstalarlos.

Un buscador de paquetes Slackware puede encontrarse en el sitio: **http://www.slackware.com/packages/**.

3.5. Otros paquetes

Además de los paquetes binarios, también existen paquetes de código fuente. Y normalmente son un paso intermedio para hacer un paquete binario desde los fuentes, en estos paquetes suele estar separado el código fuente del autor, las modificaciones hechas por el responsable del mantenimiento del paquete, los scripts a usar en la instalación y desinstalación y los datos del paquete (descripción, etc).

4. Administracion Básica del sistema

Administrar un sistema UNIX incluye muchos aspectos diferentes. Desde la instalación, hasta el manejo de seguridad que se va a utilizar. En esta sección, sin embargo, nos enfocaremos en los temas que son imprescindibles saber para cualquier administrador UNIX.

4.1. Usuarios y grupos

El manejo de usuarios en UNIX se centraliza en el archivo **/etc/passwd**. Este archivo contiene: el nombre del usuario, su clave, su número de usuario, su número de grupo principal, alguna información adicional sobre el usuario, el directorio home del usuario y el shell que debe utilizar cuando se inicia el sistema. Todos estos datos estan separados por :.

Por razones de seguridad, lo más usual es que la clave del usuario no se encuentre en este archivo, sino que esté almacenada dentro de **/etc/shadow**. En este archivo se encuentra la clave encriptada, y contiene, además, información de cuándo se vence la clave, cuando se actualizó por última vez, cada cuánto tiempo se le va a pedir al usuario que cambie su clave, etc.

Los permisos del archivo **/etc/shadow** están dados de tal forma que solamente lo puede leer o modificar el superusuario, con esto, nos aseguramos que ningún usuario pueda tener acceso a la clave de otro usuario, ni siquiera en formato encriptado.

En Debian GNU/Linux, este archivo pertenece al grupo *shadow*, de tal forma que los usuarios que pertenezcan a este grupo pueden leer el contenido del archivo. Está pensado para que el administrador admita en este grupo solamente a los usuarios que se vayan a encargar de determinada administración relacionada con usuarios.

4.1.1. Permisos Especiales

Los archivos **/etc/passwd** y **/etc/shadow** pueden ser modificados solamente por el superusuario. Sin embargo, cualquier usuario puede cambiar su contraseña mediante el comando **passwd**, que debe escribir dentro de estos archivos. Esto se debe a que este comando utiliza un permiso especial.

Si miramos los permisos de este comando (**ls -l ‘which passwd‘**), veremos que tiene una **s** en el lugar de ejecución del dueño. Este permiso se llama **setuid**. Cuando la aplicación (en este caso el comando **passwd**) se esté ejecutando, tendrá los mismos permisos que el dueño del programa.

Es decir que cuando cualquier usuario quiere cambiar su clave accede a ciertos archivos con los mismos permisos que si fuera el superusuario. Este permiso especial es un arma de doble filo, y hay que usarlo con mucha moderación. Es importante tener conciencia de cuáles son los comandos que tienen este permiso, y saber que estos comandos no tengan fallas de seguridad.

También existen permisos similares para el grupo y para el resto de los usuarios, y se llaman **setgid** y **sticky bit**.

El setgid tiene el mismo sentido que el setuid, pero se utiliza con grupos, es decir, le permite a un usuario tener los mismos permisos que el grupo al que pertenece el comando, aún cuando el usuario no pertenezca.

El **sticky bit** ya no se usa. Originalmente se utilizaba para indicarle al kernel que tenía que mantener el programa en memoria aún después de haberlo ejecutado, se usaba para programas que se ejecutaran muy seguido.

El **setgid** en directorios sirve para que todos los directorios que se creen a partir de ese pertenezcan al mismo grupo.

Para asignar estos permisos usamos **chmod u+s,g+s,o+t archivo** o con la otra sintaxis **chmod 4755 /usr/bin/passwd** (deja el comando passwd con los permisos correctos).

4.1.2. Técnicas para la administración de usuarios

Existen muchas formas distintas de administrar los usuarios de un sistema, aquí sólo hablaremos de algunas de ellas.

Para agregar un usuario, podemos simplemente agregar una línea de texto con el formato que explicamos anteriormente, al archivo **/etc/passwd**. Si estamos utilizando **/etc/shadow** para guardar las claves encriptadas, es un poco mas difícil, pero podemos copiar una línea de otro usuario y cambiar solamente el nombre de usuario. Esto nos dejaría el usuario nuevo con la contraseña del anterior, para actualizar este valor utilizamos con el comando **passwd**.

Hay que tener en cuenta que el comando **passwd** le permite al administrador cambiar la contraseña de cualquier usuario, con la sintaxis **passwd usuario**.

El manejo de grupos está centralizado en el archivo **/etc/group**, que contiene: nombre de grupo, clave (normalmente ninguna), número de grupo, usuarios.

Dentro de este archivo se configuran los grupos adicionales al principal a los que puede pertenecer un usuario (recordar que el principal está definido en /etc/passwd. También existe un **/etc/gshadow**, para guardar las claves encriptadas, pero normalmente no es utilizado.

Para ayudar al administrador suelen existir herramientas como **useradd**, **userdel**, **usermod**, **groupadd**, **groupdel**, **groupmod**. Estas herramientas permiten agregar, modificar y eliminar la información de los archivos que acabamos de comentar. Reciben por parámetro los datos que sean necesarios, para luego actualizar los archivos como corresponda.

Dado que la implementación de estos comandos varía entre las distintas distribuciones, será necesario utilizar el manual de los comandos para poder conocer el formato en que reciben los parámetros.

En Debian GNU/Linux existen los comandos **adduser**, **deluser** y **addgroup**. Que nos permiten hacer estas tareas sin recordar demasiadas opciones. Si queremos crear un usuario usaremos: **adduser usuario** y nos preguntará línea por línea los datos necesarios para agregar el usuario al sistema (clave e información adicional), luego creará el directorio home del usuario y le copiará los archivos que se encuentran dentro de **/etc/skel** al directorio home recién creado.

Si queremos agregar un grupo usamos **addgroup grupo**, y si queremos agregar un usuario a un grupo usamos **adduser user group**. Para eliminar un usuario del sistema usamos **deluser usuario**. En este último caso, hay que notar que el directorio del usuario no es borrado cuando se borra el usuario.

Un usuario conectado al sistema puede convertirse en super usuario, o en cualquier otro usuario utilizando el comando **su**. Si ejecutamos **su** sin ninguna opción nos pedirá la clave del superusuario, para poder tener sus permisos. También podemos usar **su usuario** y nos pedirá la contraseña de ese usuario para convertirnos en él.

Ésta suele ser la manera correcta de usar el usuario administrador. Es decir, ingresar al sistema con un usuario normal y cuando haya que configurar o manipular algo del sistema, ejecutar el comando su.

Existe una opción muy usada para este comando: `su - usuario`, que le indica al **su** que, no solo asuma la identidad del usuario, sino que también ejecute los scripts de inicio del usuario en quien nos estamos convirtiendo (**.profile**, **.bash_profile**, **.bashrc**, etc).

4.2. Manejo de sistemas de archivos

El manejo de los diversos sistemas de archivos (file systems) que deben montarse al inicio, o que pueden montarse más adelante, se configura en el archivo **/etc/fstab**. Este archivo contiene: nombre del dispositivo, directorio donde montarlo, tipo de sistema de archivos, opciones, configuración de backup, configuracion de verificación al inicio.

Los primeros cuatro parámetros son manejados por el comando **mount**, el quinto por el comando **dump** y el sexto por **fsck**.

Dispositivo a montar es normalmente un archivo que se encuentra dentro del directorio **/dev**, pero puede ser cualquier cosa, representación del estado del kernel (como proc), conexiones de red, imágenes ISO, etc.

Directorio donde montarlo, es el directorio, dentro del árbol general del sistema, donde se va a montar ese sistema de archivos.

Tipo de sistema de archivos, es el formato que tiene el dispositivo. Algunos de los formatos soportados son: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs, xiafs. Para obtener el listado completo de los formatos disponibles en el sistema, se debe consultar el manual del comando **mount**

Los formatos más usados son: *ext2* (típico en GNU/Linux), *vfat* (fat con nombres largos, usado en windows), *msdos* (normal de DOS) y *ext3*, *reiserfs*, *xfs*, *jfs* (sistemas de archivos con journalling, es decir, que siempre se encuentran en un estado consistente).

Opciones son muy numerosas, y es recomendable consultar el manual del comando **mount** para obtener la referencia completa. Se las escribe separadas por comas sin espacios.

Cabe destacar la opción **user** que permite que cualquier usuario pueda montar o desmontar ese sistema de archivos. Y la opción **async**, que le indica al kernel que ese sistema de archivos se puede actualizar asincrónicamente (mediante un buffer).

Backup se utiliza para especificar que sistemas de archivos debe backupear el sistema. Si la opción está en 0 no se realizará el backup. El comando **dump**, es el que se activa en caso contrario.

Verificación le indica al sistema que verifique ese sistema de archivos al inicio, utilizando el comando **fsck**. Aún si está en 0, si el sistema de archivos es de tipo **ext2**, será necesario realizar la verificación cuando no haya sido desmontado correctamente. Los números mayores a 0 le dicen al sistema el orden en que debe realizarse esta verificación.

5. Inicio del sistema

Este es un pequeño recorrido por el inicio del sistema, desde que arranca el bootloader, hasta que es posible iniciar una sesión de usuario en el sistema.

5.1. Bootloader

El bootloader es el programa que se sitúa en el sector de arranque (boot) de un disco. Es el que recibe la orden del BIOS de iniciar el sistema. Es también el encargado de ejecutar el sistema operativo que nosotros queremos usar.

En GNU/Linux los bootloaders más comunes son el **LILO** y el **GRUB**. Ambos son muy configurables, y permiten arrancar el sistema desde las más extrañas configuraciones. El caso de grub va un poco mas allá, ya que nos permite cargar el sistema desde la red, un puerto serial, etc. Además, nos puede dar una línea de comandos, de forma que podamos decirle exactamente qué hacer.

A continuación, un ejemplo de configuracion típica de LILO, del archivo **/etc/lilo.conf**.

```
lba32
boot=/dev/hda
```

```
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
delay=20
vga=normal
default=Linux

image=/vmlinuz
      initrd=/initrd.img
      label=Linux
      read-only

image=/vmlinuz.old
      initrd=/initrd.img.old
      label=LinuxOLD
      read-only
      optional

other=/dev/hda1
       label=Windoze
       restricted
```

En primer lugar, el significado de las primeras líneas del archivo:

- La línea **lba32** es necesaria si usamos rígidos IDE LBA.
- La opción **boot** indica dónde se va a instalar el sector de booteo.
- La opción **root** es el filesystem que vamos a usar como /, es decir, como directorio raíz.
- **delay** es el tiempo que va a esperar al usuario, para que ingrese algún valor distinto del default. Pasado ese tiempo, se iniciará el sistema operativo indicado como default.
- Con la opción **vga**, podemos decirl que modo de video usar. Es decir que, el lilo puede iniciar el sistema en modo gráfico (siempre que tenga soporte de nuestra placa de video).
- La opcion **default** le dice cuál es la configuracion que va a usar por omisión, es decir, la que se ejecutará si no apretamos nada cuando está iniciando, y la que aparecerá seleccionada al cargarse el Lilo.

Cada bloque **image=** y **other=** definen una forma de arramque. Cada una de estas debe tener un **label** (etiqueta), que será la forma de identificarlos, tanto dentro de la configuración del Lilo, como para el usuario en el momento de elegir qué sistema ejecutar.

Las líneas de **image** indican cuál es el kernel que debe utilizarse, y de ser necesario también debe especificarse un initrd.

La líneas **other** definen otros sistemas operativos. Normalmente, este tipo de configuración es utilizada por los sistemas que no están pensados para que se les pueda cambiar el kernel fácilmente.

Una configuracion tipica dentro de grub seria:

```
timeout 10
default 0

title GNU amadeus 2.4.19
root (hd0,0)
kernel /boot/vmlinuz-2.4.19 root=/dev/hda1

title GNU amadeus 2.4.19
root (hd0,0)
kernel /boot/vmlinuz-2.4.19 root=/dev/hda1 ro single

title GNU system Debian (kernel 2.4.18-686)
root (hd0,0)
kernel /boot/vmlinuz-2.4.18-686 root=/dev/hda1
initrd /boot/initrd.img-2.4.18-686

title GNU system Debian - single-user (kernel 2.4.18-686)
root (hd0,0)
kernel /boot/vmlinuz-2.4.18-686 root=/dev/hda1 ro single
initrd /boot/initrd.img-2.4.18-686
```

Como podemos ver, la especificación del sistema de archivos raíz es distinta de la de LILO, y de Linux en general. En *(hd0, 0)* hd0 representa el primer disco rígido y 0, la primera partición..

Al arrancar con grub es posible elegir una de estas opciones, cambiar estas configuraciones o ir directamente a una línea de comandos en la cual se puede autocompletar con tab y escribir fácilmente comandos de grub.

Podemos observar en este tipo de configuración, que muchas de las líneas de configuración de LILO son en realidad parámetros que recibe el kernel. El sistema de archivos raíz es un parámetro imprescindible.

5.2. Kernel

Como muchas veces hemos hecho notar, **Linux** es el nombre del kernel del sistema operativo que estamos utilizando. De forma que en esta sección, cada vez que utilicemos el término **Linux** nos estaremos refiriendo al kernel.

El bootloader le pasa el poder a Linux. Y le envía algunos parámetros imprescindibles, como el sistema de archivos raíz. El kernel puede recibir muchos otros parámetros, que se utilizan para configurar el soporte de hardware que está compilado en el kernel, especificar la forma de arranque, etc.

Por ejemplo, existe un parámetro **sb** que permite configurar la placa de sonido sb, solamente si el soporte está compilado dentro del kernel (que no es lo mismo que tenerlo en un módulo).

También existe un parámetro **nfsroot** con el cual le podemos indicar que el sistema de archivos raíz debe ser leído en un determinado servidor dentro de la red. Este parámetro se utiliza en máquinas que tienen un disco rígido muy pequeño, o que no tiene disco rígido.

Un parámetro para destacar es **init**. Este es el primer programa que se ejecuta una vez cargado el kernel, y se ejecuta en *modo usuario*. Normalmente este valor no lo vamos a cambiar. El comando que se utiliza por lo común es **/sbin/init**, que realiza muy bien la tarea para la cual fue pensado. Pero bajo determinadas situaciones podemos querer reemplazarlo.

Por ejemplo, si estamos arrancando la máquina, y le pasamos al kernel como parámetro `init=/bin/bash`, podremos tener acceso directo a una línea de comandos sin tener que ingresar a la máquina, y con permisos de superusuario.

Esto es, por supuesto, algo que un administrador de sistema puede querer evitar, porque implica que una persona con acceso físico a la computadora podría montar el sistema de archivos y tener control total de la computadora. Para evitarlo, será necesario editar la configuración del bootloader para que no permita hacer esto.

Para más información revisar Documentation/kernel-parameters.txt dentro de los fuentes del kernel.

5.3. init

Una vez que se ha cargado en memoria, el kernel ejecuta su primer programa **init**. Esta aplicación va a ser el *padre* de todos los procesos del sistema.

El comando **/sbin/init** utiliza la configuración que se encuentra en el archivo **/etc/inittab**, primero en búsqueda de **initdefault**, que le indica cuál es el *runlevel* en el que debe entrar.

Un **runlevel** es una configuración que permite iniciar cierto grupo de programas. Es decir, permite armar un grupo de programas que se deben ejecutar uno a continuación del otro, para realizar una tarea en particular.

Típicamente en UNIX existen 7 runlevels diferentes, el 0 halt, 1-5 configurables por el administrador y 6 reboot. En GNU/Linux suele existir, además, un runlevel S o s, para ser usado en tareas de rescate y los runlevels 7-9 que normalmente no se usan.

Si el **/etc/inittab** no existe o no tiene una línea que contenga el *initdefault* configurado, el comando **/sbin/init** pedirá al usuario el runlevel deseado. También se puede pasar el runlevel con el cual se quiere iniciar como un parámetro al kernel.

La sintaxis del archivo **/etc/inittab** es bastante simple y está extensamente explicada en el manual **man inittab**. Cada línea contiene id:runlevels:action:process.

Ejemplo:

```
id:2:initdefault:
si::sysinit:/etc/init.d/rcS
~~:S:wait:/sbin/sulogin
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
z6:6:respawn:/sbin/sulogin
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let this work."
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
```

5.4. Scripts de inicio

Cuando se inicia la máquina, se suelen ejecutar una serie de scripts básicos del sistema. En Debian GNU/Linux estos scripts son **/etc/init.d/rcS** y **/etc/init.d/rc**, este último recibe como parámetro el runlevel con el que debe iniciar.

Estos scripts ejecutan otros scripts, que se encuentran en **/etc/rcS.d** y **/etc/rc[1-6].d**, respectivamente. Que a su vez son symlinks de scripts que estan en **/etc/init.d/**.

Los symlinks tienen la forma [SK][0-9][0-9]nombre, la S o la K le dice si debe iniciar o detener ese proceso respectivamente. Esto lo hace ejecutando el script con el parámetro *start* o *stop* según corresponda. El número de 00-99 determina el orden de ejecución.

Los scripts que se encuentran en **/etc/rcS.d** hacen varias comprobaciones y múltiples cosas para dejar el sistema funcionando. Se considera que hasta este momento solo está accesible el sistema de archivos raíz y algún script de **/etc/rcS.d** se encarga de montar los otros sistemas de archivos.

Luego de dejar el sistema mínimamente funcional, los scripts de cada runlevel iniciarán los servicios que sean necesarios. Como el apache (web server) o el gdm (logueo al sistema en modo gráfico).

5.5. Ingreso al sistema

El archivo **/etc/inittab** define qué y cuántas consolas virtuales tendremos y qué programa se utilizará para ingresar al sistema. Además, también puede haber algún programa que se cargue en un script como un demonio (por ejemplo, el **gdm**).

Algunos otros servicios (como telnet, o ssh) pueden ofrecer una forma alternativa de ingresar al sistema.

5.6. Servicios / Daemons

Entre los servicios que se inician al ejecutarse algún runlevel, existen algunos que podemos esperar que se encuentren en cualquier sistema UNIX. Como el **syslog**, el **cron**, **anacron**, **at** y **inetd**.

El **syslog** es el responsable de distribuir los mensajes del sistema en distintos archivos de log. La configuración se encuentra en **/etc/syslog.conf**, y la mayoría de los logs se almacenan en el directorio **/var/log**.

Un usuario puede mandar mensajes al log del sistema con el commando **logger**.

Los servicios **cron**, **anacron** y **at** se encargan de ejecutar tareas en determinados momentos. Pero cada uno de ellos, lo hace de una manera distinta.

cron se encarga de ejecutar tareas en determinado momento. El archivo de configuracion típico es **/etc/crontab** y su sintaxis es: minutos hora diadelmes mes diadelasemana usuario comando

Nuevas versiones de **cron** tienen además un directorio **/etc/cron.d**, que contiene varios archivos con el mismo formato que el crontab. Esta separación en archivos se utiliza para tener tareas programadas separadas por grupos.

anacron ejecuta un comando, sólo si hace una determinada cantidad de días que no se ejecuta. A diferencia de **cron**, no asume que el sistema va a estar prendido todos los días las 24 hs. El archivo de configuracion es **/etc/anacrontab** y sintaxis: dias retraso identificador comando.

El directorio **/var/spool/anacron** suele tener archivos con la información necesaria para utilizar **anacron**.

atd es el servicio que recibe pedidos desde el comando **at**. Es una cola de procesos que se ejecutarán en orden cronológico.

El comando **at** puede ser utilizado por cualquier usuario (excepto aquellos que se encuentren en el archivo **/etc/at.deny**) y el único parámetro necesario es la hora en que se debe ejecutar el comando, el comando a ejecutar se lee por entrada estándar.

Con **atq** se puede revisar la cola de procesos programados y con **atrm** quitar un proceso de la cola.

Por otro lado, tenemos al **inetd** que es un superdemonio que se encarga de recibir y derivar servicios de red. Su archivo de configuración es **/etc/inetd.conf** y tiene la siguiente sintaxis: nombredelservicio tipodeconexión protocolo flags usuario servicio parámetros.

Utilizando el nombre del servicio, se puede encontrar en el archivo **/etc/services** el un número de puerto correspondiente. El **inetd** se quedará esperando recibir una conexión con ese protocolo, en ese puerto, y una vez iniciada la conexión, le desviará a otro comando esa conexion.