1 Instalación e Compilación de software

1.1 Sumario

1.2 Instalación de software en GNU/Linux

La instalación y mantenimiento de versiones del software de un sistema operativo es uno de los aspectos fundamentales de la administración de un sistema informático. En GNU/Linux existen una gran cantidad de herramientas disponibles para este cometido, desde interfaces gráficas, hasta herramientas de línea de comandos. Como siempre, el utilizar la linea de comandos, constituye un entorno más flexible y estándar que las herramientas basadas en GUI.

Existen varias herramientas y comandos para gestionar la instalación de programas en entornos Linux. Dependiendo del tipo de distribución Linux de que se trate encontramos diferencias a la hora de gestionar el software del Sistema Operativo.

El software en los sistemas Linux se organiza en paquetes que incluyen la versión binaria del programa o utilidad. La idea de este esquema es poder descargar el paquete e integrarlo en nuestro sistema del modo más directo posible. Todos los sistemas incluyen algún tipo de registro de orígenes de paquetes (repositorios). La gestión de repositorios es una de las tareas de administración importantes en la administración del software del sistema. Por ejemplo, en Debian y derivados, los repositorios están registrados en el archivo de texto /etc/apt/sources.list y en los archivos dentro del directorio /etc/apt/sources.list.d.

Por ejemplo, en los sistemas basados en Debian (como Ubuntu) se utiliza el formato de paquetes binarios .deb. Otros sistemas, como RedHat, Fedora o CentOS utilizan formato de paquetes binario .rpm. Antiguamente se gestionaba de modo manual la instalación de paquetes de software en el sistema, sin embargo esta estrategia causaba múltiples problemas cuando había dependencias involucradas, es decir, cuando ese paquete dependía para su funcionamiento de otros paquetes que no estaban instalados en el sistema. Para solucionar este problema se desarrollaron herramientas que gestionan automáticamente las dependencias de los paquetes, de modo que, el usuario simplemente introduce la orden de instalación del paquete correspondiente y los paquetes de los que depende se instalan automáticamente.

1.2.1 Repositorios

Los repositorios son los almacenes de software, comúnmente denominados paquetes, de las distribuciones Linux. Los repositorios son accesibles públicamente a través de protocolos estándar como http, ftp o https. Utilizando repositorios eliminamos la necesidad de disponer de los binarios de cada una de las utilidades que vamos a necesitar en nuestro sistema. De modo que, si es necesario algún paquete procedemos a instalarlo directamente descargándolo desde el repositorio y ejecutando un script de instalación. Este proceso está automatizado a través de herramientas como las que veremos más adelante.

Nuestro sistema GNU/Linux, en función de su tipo, utilizará un formato de paquete determinado, siendo los más habituales deb y rpm. En el caso de Debian y derivados se utiliza paquetería deb. Para descargar e instalar la versión binario de un paquete, en formato deb, o rpm, podríamos descargarlo

e instalarlo directamente. Sin embargo es común que se produzcan conflictos relacionados con las dependencias, es decir, el conjunto de paquetes necesarios para que el que queremos instalar pueda funcionar. Para evitar los problemas derivados de este hecho se utilizan herramientas que gestionan automáticamente estas dependencias.

La lista de repositorios, como vimos en el apartado anterior, se definen en el archivo **/etc/apt/sources.list** y dentro del directorio **/etc/apt/sources.list.d**. Es importante disponer el menos de una lista mínima de repositorios de la distribución que estemos utilizando. Repositorios mínimos para Debian 9 (stretch)

En **/etc/apt/sources.list** deberíamos tener al menos

```
#Repositorios relacionados con las actualizaciones de seguridad
deb http://security.debian.org/debian-security stretch/updates main contrib
deb-src http://security.debian.org/debian-security stretch/updates main contrib

#Repositorios de paquetería Debian
deb http://deb.debian.org/debian/ stretch-updates main contrib
deb http://deb.debian.org/debian/ stretch main contrib

#Repositorios de versiones de código fuente de los paquetes
deb-src http://deb.debian.org/debian/ stretch-updates main contrib
deb-src http://deb.debian.org/debian/ stretch main contrib
```

Cada una de las líneas anteriores consta de

- **Formato**: en este caso paquetes deb
- **URL** de localización del repositorio
- **Versión** de la distribución: en este caso stretch
- **Ramas**: dentro de las cuales tenemos
    - **main**: rama principal de paquetes de la distribución. Contiene el 90% de los paquetes de la distribución. La totalidad de estos paquetes son libres y cumplen con los 10 principios de la Debian Free software Guidelines.
    - **contrib**: rama de paquetes de colaboradores de la distribución. Contiene software que también cumple con los principios de la Debian Free software Guidelines, pero que precisa de software o paquetes no libres para su compilación y/o ejecución.
    - **non-free**: rama de paquetes de código privativo asociados a la distribución. Incluye paquetes privativos que obviamente no cumplen con las directrices de la Debian Free software Guidelines.

Una vez definidos los repositorios tendremos que actualizar nuestra caché local de paquetes, consistente en el nombre del paquete, la URL del repositorio en el que se encuentra, y la versión del mismo. Para ello usaremos alguna de las herramientas que se explicarán a continuación.

1.2.1.1 Repositorios https

Es posible utilizar repositorios https en los archivos de registro de los mismos, como /etc/apt/sources.list. Para ello previamente debemos instalar un paquete que nos permita conectarnos a repositorios que utilizan este protocolo seguro

```
apt install apt-transport-https
```

Estos repositorios requerirán procedimiento de autenticación y por tanto deberemos disponer las claves públicas correspondientes para poder utilizarlos

1.2.1.2 Claves GPG de autenticación en repositorios

Algunos repositorios, http o https, necesitan de acreditar su identidad para poder descargar paquetes desde ellos. Este es un mecanismo de seguridad que evita ataques de suplantación de identidad del repositorio. De este modo, cada vez que nos conectemos al repositorio tendremos la garantía de que su identidad es legítima. Para eso necesitamos disponer de la correspondiente clave pública del repositorio. En caso de no disponer de la clave obtendremos un error como el siguiente a la hora de ejecutar comando apt

```
W: Error de GPG: https://dl.ring.cx/ring-nightly/ubuntu_18.04 ring InRelease: Las firmas siguientes no se pudieron verificar porque
```

Para solucionarlo incorporamos la clave apt con el comando **apt-key**

1.2.1.2.1 Ejemplo: Incorporar un repositorio con clave GPG

Vamos a incorporar el repositorio para instalar ring en el directorio de /etc/apt/sources.list.d

```
sh -c "echo 'deb https://dl.ring.cx/ring-nightly/ubuntu_18.04/ ring main' > /etc/apt/sources.list.d/ring-nightly-main.list"
```

Si ejecutamos

```
apt update
```

Nos encontraremos con el error mencionado. Por tanto, para solucionarlo, vamos a ejecutar el comando para añadir la clave GPG del repositorio

```
apt-key adv --keyserver pgp.mit.edu --recv-keys 64CD5FA175348F84
```

El último argumento del comando, a continuación de --recv-keys es el Key ID de la clave buscada. Ahora sí ya podremos utilizar el repositorio y todos los paquetes disponibles en él.

NOTALa misma acción podría haberse realizado en varios comandos, primero recibimos del repositorio:

```
gpg --keyserver pgp.mit.edu --recv 64CD5FA175348F84
```

a continuación exportamos la clave para añadirla al llavero de apt:

```
gpg --export --armor 64CD5FA175348F84 | apt-key add -
```

Si obtenemos un error relacionado con la utilidad dirmngr la instalamos

```
apt install dirmngr
```

1.2.2 Repositorios PPA (Personal Package Archive)

Los repositorios PPA son repositorios que permiten a los desarrolladores distribuir software y actualizaciones de forma directa, sin necesidad de que éstos estén disponibles en los repositorios oficiales. Ya sabemos que los repositorios oficiales soportan determinadas versiones de los paquetes que no siempre pueden coincidir con nuestras necesidades.

Es común que los desarrolladores utilicen estos repositorios para distribuir sus propias versiones y actualizaciones de los paquetes que desarrollan. Para incorporar un repositorio de este tipo a nuestro sistema podemos utilizar el comando **add-apt-repository**, disponible en el paquete **software-properties-common**. En caso de no estar instalado lo hacemos con el siguiente comando

```
apt install software-properties-common
```

Una vez instalado ya podremos incorporar estos repositorios al sistema e instalar los paquetes disponibles en ellos

1.2.2.1 Ejemplo de instalación de paquete a partir de repositorio ppa

Vamos a instalar **el entorno JDK de Java en su versión 8**, para ello usaremos un repositorio ppa

```
add-apt-repository ppa:webupd8team/java
apt update
apt install oracle-java8-installer
```

Para eliminar un repositorio ppa que no vamos a utilizar o que no nos interesa conservar ejecutamos el mismo comando con la opción --remove:

```
add-apt-repository --remove ppa:webupd8team/java
```

1.2.3 Comandos apt-get y apt-cache

APT, Advanced Package Tool, es un conjunto de comandos y herramientas utilizados en sistemas Debian para gestionar su software. Algunos comandos útiles:

Es un comando con varios opciones que determinan su funcionamiento:

- **apt-get**
  - **apt-get update**: Actualiza la caché de paquetes del sistema con las últimas versiones de los repositorios
  - **apt-get upgrade**: Actualiza los paquetes instalados
  - **apt-get dist-upgrade**: Actualiza la totalidad de los paquetes instalados
  - **apt-get install paquete**: Instala el paquete indicado en el sistema. Para instalar una versión concreta de un paquete, podemos utilizar la opción paquete=version

- **apt-get download paquete**: Descarga el paquete binario en el directorio actual
- **apt-get check paquete**: Comprueba si hay dependencias rotas
- **apt-get source paquete**: Descarga fuentes del paquete
- **apt-get build-dep paquete**: Configura dependencias de compilación para paquetes fuente
- **apt-get install -f paquete**: Instala el paquete resolviendo dependencias incumplidas
- **apt-get remove paquete**: Elimina el paquete indicado del sistema
- **apt-get autoremove**: Elimina paquetes que no son necesarios por ser depedencias no usadas de paquetes ya desintalados
- **apt-get purge paquete**: Elimina el paquete indicado del sistema y los archivos de configuración asociados
- **apt-get autoclean**: Elimina los .deb de versiones de paquetes antiguas de la caché en /var/cache/apt/archives
- **apt-get clean**: Limpia los archivos de paquetes de la caché, antiguos o no

Es buena práctica, sobretodo antes de un upgrade, o dist-upgrade, el ejecutar un apt-get update para actualizar las cachés de paquetes locales

- **apt-cache**: Se utiliza para gestionar la caché de paquetes del sistema
  - **apt-cache search paquete**: Busca información sobre el paquete indicado en la caché del sistema). Pueden utilizarse comodines y expresiones regulares para contextualizar la búsqueda
  - **apt-cache depends paquete**: Muestra las dependencias del paquete indicado
  - **apt-cache stats**: Muestra estadísticas sobre paquetes que hay en la caché
  - **apt-cache unmet**: Muestra dependencias incumplidas de paquetes
  - **apt-cache policy paquete**: Muestra las versiones instaladas y disponibles en el repositorio de un paquete determinado

1.2.4 Comando apt

De uso muy similar, prácticamente idéntico, a apt-get, permite una mayor usabilidad, simplicidad y claridad de información. **Es la herramienta de gestión de paquetes recomendada por las distribuciones más recientes** basadas en paquetería debian

Algunos de los comandos apt más utilizados:

- **apt search paquete**: Busca un paquete en la caché, al estilo de apt-cache search
- **apt list**: Muestra listado de paquetes
- **apt show paquete**: Muestra información del paquete indicado
- **apt update**: Actualiza la caché de paquetes del sistema con las últimas versiones de los repositorios
- **apt upgrade**: Actualiza el sistema instalando y actualizando paquetes
- **apt full-upgrade**: Actualiza el sistema borrando, instalando y actualizando la totalidad de los paquetes. La diferencia principal con la opción **upgrade** es que con **full-upgrade** serán eliminados aquellos paquetes que es necesario borrar para completar la actualización, algo que upgrade no hace.
- **apt install paquete**: Instala el paquete indicado en el sistema
- **apt remove paquete**: Elimina el paquete indicado del sistema
- **apt autoremove**: Elimina paquetes que no son necesarios por ser dependencias no usadas de paquetes ya desintalados
- **apt source paquete**: Descarga fuentes del paquete
- **apt build-dep paquete**: Configura dependencias de compilación para paquetes fuente
- **apt purge paquete**: Elimina el paquete indicado del sistema y los archivos de configuración asociados
- **apt autoclean**: Elimina los .deb de versiones de paquetes antiguas de la caché en **/var/cache/apt/archives**
- **apt clean**: Limpia los archivos de listas de paquetes de la caché, antiguos o no
- **apt policy paquete**: Muestra la versión instalada y las disponibles en repositorio para un paquete dado
- **apt edit-sources**: Permita editar las fuentes de repositorios

1.2.5 Instalando con apt

Vamos a ver un pequeño supuesto relacionado con el comando apt a partir de una serie de pasos

- **Actualizar la caché de paquetes**: Para ello, una vez que tenemos los repositorios definidos, el primer paso será actualizar la caché local de paquetes de la distribución

```
apt update
```

- **Mostrar un listado de paquetes**: Podemos usar el comando

```
apt list
```

el resultado del comando anterior será una lista extensa, podemos restringir los resultados mostrados filtrando con grep

```
apt list | grep mariadb
```

mostraría aquellos paquetes que contienen el texto **mariadb** en su nombre o descripción

- **Mostrar información del paquete build-essential**: Ejecutamos para ello

```
apt show build-essential
```

- **Actualización completa de los paquetes del sistema**: Usaremos para ello el comando **apt full-upgrade**, sin embargo para evitar realizar la actualización de modo efectivo lo vamos a ejecutar de modo simulado, con la opción **-s**, la cual nos mostrará simplemente los paquetes que se actualizarán y el tamaño de la descarga

```
apt full-upgrade -s
```

- **Buscar paquetes relacionados con nginx**: Ejecutando

```
apt search nginx
```

- **Por último vamos a instalar el paquete software-properties-common**

```
apt install -y software-properties-common
```

1.2.6 Herramienta Aptitude

Interfaz de gestión de paquetes muy funcional y cómoda, al estilo de apt-get, pero más coherente. Comandos:

- **aptitude**: Al ejecutarlo sin argumentos muestra una interfaz para buscar, navegar, instalar, actualizar y realizar otras tareas de administración de paquetes.
- **aptitude install paquete**: Instala software en tu sistema, junto con las dependencias necesarias.
- **aptitude remove paquete**: Elimina paquetes junto con las dependencias que queden huérfanas.
- **aptitude purge paquete**: Elimina paquetes y dependencias huérfanas junto con los ficheros de configuración.
- **aptitude search paquete**: Busca paquetes en las listas de paquetes locales de apt.
- **aptitude update**: Actualiza las listas de paquetes locales.
- **aptitude safe-upgrade**: Actualiza los paquetes disponibles de modo seguro, sin borrar ninguno de los existentes
- **aptitude full-upgrade**: Actualiza los paquetes disponibles de modo seguro, borra si es necesario algún paquete existente para actualizar otro
- **aptitude clean**: Elimina los ficheros que fué necesario descargar para instalar software en tu sistema.
- **aptitude show paquete**: Muestra detalles acerca del paquete nombrado.
- **aptitude autoclean**: Elimina los paquetes deb obsoletos.
- **aptitude hold paquete**: Fuerza a que un paquete permanezca en su versión actual, y no se actualice.

Con la opción -s, al invocar alguno de los comandos anteriores se ejecuta la acción pero sin descargar el paquete y modificar el sistema, solo simula la acción

1.2.7 Comando dpkg

Este comando se utiliza para gestionar directamente los paquetes .deb. A diferencia de apt-get no gestiona automáticamente las dependencias.

Para instalar un paquete en Debian o Ubuntu con este comando utilizamos la sintaxis:

dpkg -i paquete.deb

Para borrar un paquete podemos utilizar la opción -r, con la sintaxis:

dpkg -r paquete

La opción --purge elimina también todos los archivos de configuración asociados al paquete.Veamos un ejemplo:

Borrado de toda la información de configuración de paquetes eliminados

```
dpkg -l | grep '^rc' | awk '{print $2}' | xargs dpkg --purge
```

Para lanzar las opciones de configuración de un paquete que ya ha sido instalado podemos ejecutar:

dpkg-reconfigure paquete

1.3 Parámetrización del Kernel y Compilación de Software en GNU/Linux

1.3.1 Parámetros del kernel

Es posible modificar ciertos variables de parámetros del kernel que alterarán el comportamiento del sistema. Para ello debemos editar alguno de los siguientes componentes

- Archivo /**etc**/**sysctl.conf**: Algunas variables, relacionadas con IPv4 e IPv6 se establecen directamente en este archivo de texto. Cada una está perfectamente definida y comentada en el archivo
- **Directorio** /**etc**/**sysctl.d**: Otras variables pueden ser configuradas dentro de los archivos de este directorio, los cuales contienen variables de kernel para distintos aspectos

Utilizando el comando **sysctl** se permite visualizar información y configurar los parámetros de tiempo de ejecución del kernel, para ello especificaremos distintos valores de variables del kernel mediante la sintaxis **variable=valor**

```
sysctl -w kernel.hostname=miequipo
```

La opción -w (write) especifica la modificación (escritura) de la correspondiente variable.

Para ver un listado con variables y valores usamos la opción -a

```
sysctl -a
```

Para cargar en sysctl los valores de variables definidos en sysctl.conf usamos la opción -p

```
sysctl -p
```

Los parámetros que se pueden modificar se encuentran en /proc/sys, dentro de este directorio encontramos los siguientes: abi, debug, dev, fs, kernel, net, vm, dentro de los cuales tenemos archivos y directorios que estructuran las distintas variables en función de su contexto.

1.4 Compilación de software

Para descargar archivos de Internet mediante línea de comandos, por ejemplo para descargar las fuentes de un programa que queremos compilar o instalar existe un comando, **wget**, que permite descargar archivos de servidores HTTP o FTP

```
wget ftp://gnjilux.cc.fer.hr/welcome.msg
wget http://fly.cc.fer.hr/welcome.msg
```

Hay varios opciones soportadas pero en o esencial descarga el archivo apuntado por la URL pasada como parámetro

1.4.1 Compilación desde las fuentes

En ocasiones necesitaremos construir los programas directamente desde el código fuente (recordad que Linux es software libre). En general será más habitual encontrar un binario adecuado para la distribución Linux que estemos utilizando, en especial para distribuciones comunes, como Ubuntu, Debian, OpenSuse, etc.

Aún así a veces, por la rareza del propio software a instalar, o por la versión en concreto, o simplemente por cuestiones de querer optimizar el software para nuestra plataforma y sistema en concreto, se requiere realizar el proceso de compilación desde las fuentes para su posterior instalación.

Previamente deberemos de disponer de todas las herramientas necesarias para la compilación de programas. Consistentes básicamente en un compilador de C (programa gcc, g++) y librerías. Con el siguiente comando instalamos un meta-paquete (que contiene otros paquetes) que instalará en el sistema todo lo necesario para realizar el proceso de compilación

```
apt install build-essential
```

Los pasos a dar son los siguientes:

- **Descarga de los fuentes**: Habitualmente en formato tar.gz, tar.zip, tar.bz2. Usando wget y la URL al archivo podemos descargarnos el tar.bz2 o tar.gz
- **Descomprimir las fuentes** con el comando tar

```
tar -xzvf nombre_del_programa.tar.gz
```

(Para archivos .bz2 en lugar de la opción z usaremos la opción j).

Leer el archivo **README** dentro del directorio resultado del proceso de descompresión anterior. **Leer el archivo README (como su nombre indica) es muy importante**, pues no siempre el proceso de configuración y compilación que veremos a continuación se realiza del mismo modo

- **Comprobar si hay dependencias** que el programa necesite:

```
apt build-dep nombre_del_programa
```

- **Ejecutar ./configure** para preparar las fuentes para nuestra plataforma:

```
./configure
```

En la llamada al ./configure anterior pueden pasarse múltiples opciones de configuración, previos a la compilación, que personalizan el comportamiento del proceso de creación del binario durante el posterior proceso de compilación. Una de las opciones más habituales es **--prefix** el cual permite establecer el directorio en el que se va a ubicar el binario correspondiente.

- **Generar el binario (compilación)**

```
make
```

- **Instalar**

```
make install
```

...para desinstalar:

```
make uninstall
```

1.4.2 Utilidad checkinstall

Cuando se compila software desde las fuentes puede ser complicado borrarlo con posterioridad a la instalación. En ocasiones los programadores crea un target **make uninstall**, pero éste no se encuentra disponible en muchos casos. Para poder hacer una instalación limpia, y que más tarde pueda ser eliminada, podemos utilizar el paquete **checkinstall**.

Este paquete consiste en una utilidad de línea de comandos checkinstall, que deberá ser **invocada tras el ./configure**. Automáticamente invoca a make y realiza un inventario de todas las modificaciones que se realizarán en el sistema. Por lo general crea un paquete .deb o .rpm desde el cual se realiza la instalación. De este modo el paquete puede ser desinstalado fácilmanente mediante, por ejemplo, un comando dpkg -r <paquete>.

Su instalación es muy sencilla

```
apt install checkinstall
```

1.4.3 Ejemplo. Compilación de nginx

Vamos a instalar el paquete nginx desde utilizando para ello el código fuente.

- En primer lugar instalamos las dependencias necesarias para poder compilar nginx desde las fuentes

```
apt build-dep nginx
```

- Ahora descargamos las fuentes de nginx en el directorio /tmp

```
cd /tmp
apt source nginx
```

- A continuación instalamos nginx desde las fuentes. En primer lugar nos movemos al directorio /tmp/nginx-<version>, donde version indica la versión de las fuentes que hemos descargado.

```
cd nginx-1.10.3
./configure
make
make install
```

Tras lo cual ya dispondremos del servidor instalado. Arrancamos el servicio

```
/usr/local/nginx/sbin/nginx
```

Y ya podremos acceder al servicio en el puerto 80 para comprobar que efectivamente funciona

1.5 Instalando módulos dinámicos

La herramienta **dkms** (dynamic kernel module support) permite instalar módulos que se vinculan al kernel durante el arranque del sistema. Permite vincular módulos al kernel que residen en árboles de código fuente diferentes al árbol de códigos fuente del kernel. Además, este aspecto se gestiona de manera automática en cada arranque. Si se detecta un cambio en el kernel el módulo se compila para la versión correspondiente del mismo.

Tradicionalmente los módulos del kernel se añadían de forma estática al mismo, de modo que, cuando se actualizaba éste a una nueva versión, los módulos desplegados de este modo dejaban de funcionar en el nuevo kernel.

Con dkms vinculamos los módulos al kernel de modo dinámico. Cuando el kernel se actualice los correspondientes módulos serán compilados e integrados en el nuevo kernel. Esta herramienta resulta muy útil para conseguir que nuestros dispositivos funcionen independientemente de la versión del kernel que estemos ejecutando.

Muchos de los módulos disponibles para Linux incorporan versiones para dkms.

1.5.1 Comandos útiles de dkms

- **dkms status:** muestra el estado de los módulos gestionados con dkms
- **dkms install:** permite instalar un módulo con dkms
- **dkms autoinstall:** actualiza todos los módulos a la nueva versión del kernel
- **dkms remove:** elimina un módulo de dkms

1.5.2 Ejemplo de instalación de módulo fuente con dkms

Primero instalamos la herramienta dkms

```
apt install -y linux-headers-`cat /proc/version | awk '{print $3}'`
apt install -y dkms
```

Antes, como vemos debemos instalar el paquete **linux-headers** para la versión correspodiente de nuestro kernel

Ahora descargamos los fuentes del módulo en el directorio tmp

```
cd /tmp
wget https://sourceforge.net/projects/e1000/files/ixgbe%20stable/5.2.4/ixgbe-5.2.4.tar.gz https://sourceforge.net/projects/e1000/fil
```

NOTA: Si la utilidad wget no está instalada en nuestro sistema, podemos instalarla con

```
apt install wget
```

Extraemos los contenidos del tar.gz y los colocamos en el directorio de fuentes del kernel

```
tar xzf ixgbe-5.2.4.tar.gz -C /usr/local/src
mv /usr/local/src/ixgbe-5.2.4/src /usr/src/ixgbe-5.2.4
```

Creamos el archivo de configuración dkms

```
vi /usr/src/ixgbe-5.2.4/dkms.conf
```

Añadimos lo siguiente a ese archivo

```
PACKAGE_NAME="ixgbe"
PACKAGE_VERSION="5.2.4"
BUILT_MODULE_NAME[0]="ixgbe"
DEST_MODULE_LOCATION[0]="/kernel/drivers/net/ethernet/intel/ixgbe/"
AUTOINSTALL="yes"
```

Añadimos a continuación el módulo a dkms

```
dkms add -m ixgbe -v 5.2.4
```

Construímos el módulo con soporte dkms

```
dkms build -m ixgbe -v 5.2.4
```

Y por último instalamos el módulo para nuestra versión del kernel

```
dkms install -m ixgbe -v 5.2.4
```

Para comprobar que efectivamente el módulo se ha instalado correctamente

```
dkms status
```

Deberíamos ver una salida como

```
ixgbe, 5.2.4, 4.9.0-3-amd64, x86_64: installed
```

A partir de ahora, cada vez que se actualice la versión del kernel, o se invoque a dkms con la opción autoinstall, se compilará de nuevo el módulo y se vinculará automáticamente con el kernel correspondiente.

Volver

JavierFP 16:58 22 nov 2017 (CET)