Tema 8: Configuración avanzada del sistema operativo

1. Parámetros de configuración

1.1 Introducción

Durante la instalación y configuración de computadores y periféricos puede ser necesario conocer o modificar algunos parámetros de configuración del sistema como:

- Dispositivos de hardware.
- Software del sistema operativo.
- Software de las aplicaciones instaladas.
- Usuarios.
- Preferencias.

La forma de guardar y acceder a estos parámetros es distinta en cada sistema operativo. En los siguientes apartados se describen los enfoques seguidos por Windows y Linux.

1.2 El Registro de Windows

Hasta la aparición de Windows 95, la forma habitual de almacenar los parámetros del sistema era en archivos **INI**: archivos de texto con una estructura muy simple, dividida en secciones (marcadas por corchetes), en las que se asignaban valores a parámetros. Estos archivos tenían varios inconvenientes:

- Sólo permiten dos niveles de anidamiento (secciones y parámetros).
- No es fácil utilizar datos binarios.
- La configuración está diseminada por muchos archivos.

El Registro de Windows es una base de datos pensada para almacenar toda la información de configuración del equipo. En principio, no está pensado para que el usuario acceda directamente a estos valores, sino que se modificarán a través de programas. Las aplicaciones pueden utilizar una API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones) para almacenar y guardar sus valores de configuración. Por ejemplo, cuando se modifica una opción de configuración del Explorer, este utiliza una función de la API para cambiar un valor en el Registro.

Los datos del registro se introducen y se usan de la siguiente manera:

- Durante la fase de instalación del sistema operativo, el programa de instalación crea el registro. Los programas de instalación de las aplicaciones pueden añadir nueva información más tarde. También se utiliza la información del registro durante la instalación para determinar la configuración del sistema.

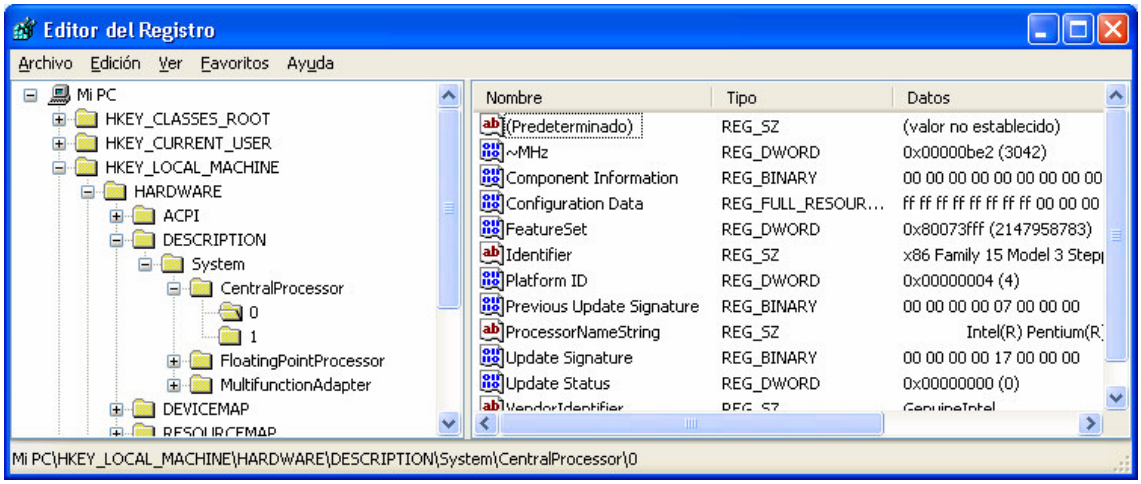
- Cada vez que se arranca el sistema, tanto el reconocedor de hardware como el núcleo del sistema operativo almacenan información sobre los dispositivos.
- El núcleo utiliza la información del Registro para decidir qué controladores de dispositivo se deben cargar durante el arranque.
- Los controladores de dispositivo también guardan información sobre los dispositivos que controlan y leen parámetros de configuración del Registro.
- Distintas herramientas administrativas permiten ver y modificar valores del Registro, haciendo además una validación de los datos.
- Las aplicaciones pueden obtener información del sistema del Registro. Además, pueden almacenar y leer sus preferencias y valores de configuración, tanto durante el arranque como durante su uso normal.

Inconvenientes del Registro son:

- No es fácil modificarlo y, al mezclar información básica del sistema operativo y de muchas aplicaciones, es difícil encontrar dónde está lo que se busca cuando se desea cambiar algo.
- Si se corrompe puede dañar todo el sistema.
- Es difícil de salvar datos correspondientes a una aplicación sólo, por ejemplo, cuando se desea realizar una migración.

1.2.1 Estructura del Registro

El Registro es una estructura de datos jerárquica, tal como se muestra en la siguiente figura:



Está organizado en una serie de subárboles. Dentro de cada subárbol hay una serie de claves. Las claves pueden contener otras claves y entradas de valor, que son los elementos que realmente almacenan la información de configuración. Se puede trazar una analogía con un sistema de ficheros en el que los subárboles serían la letra de unidad, las claves serían las carpetas y las entradas de valor serían los ficheros. De hecho, se puede referenciar un elemento mediante una

ruta (*path*) como por ejemplo HKEY_LOCAL_MACHINE\SOFTWARE\Aspell\ MajorVersion.

1.2.1.1 Subárboles

Son la raíz de la jerarquía lógica en la que se organiza el Registro. Existen dos subárboles principales:

- **HKEY_LOCAL_MACHINE (HKLM):** Guarda información de la máquina, incluyendo datos del hardware, del sistema operativo y de las aplicaciones.
- **HKEY_USERS (HKU):** Guarda información de los perfiles de usuarios cargados y del perfil por defecto. Los usuarios cargados son el usuario que haya iniciado sesión interactivamente y aquellos que tengan servicios iniciados. Nótese que aquí no están los perfiles de todos los usuarios del sistema: esos perfiles residen en ficheros y sólo se cargan cuando el usuario entra en el sistema (interactivamente o a través de un servicio).

Para facilitar el acceso a la información, se definen otros tres subárboles virtuales que, en realidad, contienen información almacenada en claves de los subárboles listados anteriormente. Estos subárboles virtuales son:

- **HKEY_CURRENT_USER (HKCU):** Almacena el perfil del usuario que ha iniciado sesión de modo interactivo. Incluye variables de entorno, configuración del escritorio, conexiones de red, impresoras y preferencias para las aplicaciones. Es en realidad un alias del subárbol HKU\Id. de seguridad del usuario actual.
- **HKEY_CLASSES_ROOT (HKCR):** Contiene información de diversas tecnologías de incrustación de objetos de Microsoft, como por ejemplo OLE (*Object Linking and Embedding*), y asociación de archivos a clases. En esta clave hay una entrada si existe la entrada correspondiente en HKLM\SOFTWARE\Classes o en HCKU\SOFTWARE\Classes. En caso de existir en ambas, prevalece el valor de HKCU.
- **HKEY_CURRENT_CONFIG (HKCC):** Contiene información sobre la configuración del hardware generada dinámicamente y que no se guarda en disco. Es un alias de HKLM\SYSTEM\CurrentControlSet\Hardware Profiles\Current.

Existe otro subárbol, HKEY_PERFORMANCE_DATA, que contiene información de rendimiento del sistema y no se muestra en el editor del registro.

1.2.1.2 Claves

Las claves son un elemento de agrupación, similar a las carpetas en los sistemas de ficheros, que pueden contener otras claves y entradas de valor.

1.2.1.3 Entradas de valor

Las entradas de valor guardan los datos. Están compuestas de tres elementos: nombre, tipo y valor. El tipo es un número de 32 bits, aunque se suele hacer

referencia a los distintos tipos utilizando el nombre de la constante definida en la API de Win32. Los tipos más importantes predefinidos por el sistema son:

- **REG_BINARY:** Datos binarios sin procesar. La mayor parte de la información de los componentes de hardware se guarda en forma de datos binarios y se presenta en el Editor del Registro en formato hexadecimal.
- **REG_DWORD:** Datos representados por un número de 4 bytes de longitud. Muchos parámetros de controladores de dispositivo y de servicios son de este tipo, y se presentan en el Editor del Registro en formato binario, hexadecimal o decimal.
- **REG_SZ:** Cadena de texto de longitud fija.
- **REG_EXPAND_SZ:** Cadena de datos expansible. Puede incluir variables, como por ejemplo *%SystemRoot%*, que se resuelven cuando un programa utiliza los datos.
- **REG_MULTI_SZ:** Lista de cadenas.
- **REG_FULL_RESOURCE_DESCRIPTOR:** Serie de tablas anidadas, diseñadas para almacenar una lista de recursos para un componente de hardware o un controlador.

1.2.1.4 Secciones

En este contexto, el término sección (*hive*) describe una agrupación lógica de claves, subclaves y valores que se guardan en un mismo conjunto de archivos siguiendo este esquema:

Sección del Registro	Nombres de archivo
HKEY_LOCAL_MACHINE\SAM	Sam, Sam.log y Sam.sav
HKEY_LOCAL_MACHINE\SECURITY	Security, Security.log y Security.sav
HKEY_LOCAL_MACHINE\SOFTWARE	Software, Software.log y Software.sav
HKEY_LOCAL_MACHINE\SYSTEM	System, System.alt System.log y System.sav
HKEY_CURRENT_CONFIG	System, System.alt, System.log, System.sav, Ntuser.dat, Ntuser.log
HKEY_USERS\.DEFAULT	Default y Default.log

Los archivos *.log* son archivos en los que se van apuntando los cambios que se van a hacer antes de hacerlos, a modo de registro de transacciones, para evitar que se corrompa el archivo si hay un fallo a la mitad de una operación. Los archivos *.sav* son copias al final de la fase en modo texto del proceso de instalación, antes que comience la fase gráfica. El archivo *System.alt* es una copia de seguridad del archivo *System*. Todos los archivos se almacenan en el directorio *%SystemRoot%\System32\Config*, excepto *NTuser* y *NTuser.log*, que se guardan en el directorio del usuario.

1.2.2 Administración del registro

Windows proporciona una aplicación[1] para ver y cambiar valores del registro: el *Editor del Registro*. En principio, se desaconseja emplear esta aplicación ya que no comprueba que los cambios que se hacen sean coherentes (que los valores sean válidos, que no se borren elementos fundamentales, etc.), con lo que es mucho más fácil cometer errores, que pueden tener graves consecuencias para la estabilidad del equipo o de las aplicaciones.

Se pueden hacer copias de seguridad del registro a través de la herramienta de copia de seguridad incluida en Windows, activando la casilla *Estado del sistema*. Además, se pueden exportar claves del registro, mediante el *Editor del Registro*, a ficheros *.reg*, que son ficheros de texto con la siguiente sintaxis:

```
[<Nombre sección>\<Clave>\<Subclave>]
"Entrada de valor"=<Tipo>:<Datos>
```

1.2.3 API de acceso al registro

Existe un conjunto de funciones dentro del API de Win32 para manejar el registro. Entre las más importantes se encuentran:

- *RegOpenKeyEx()*: Abre una clave.
- *RegSetValueEx()*: Da valor a una entrada de valor.
- *RegCloseKey()*: Cierra una clave.
- *RegDeleteKey()*: Borra una clave.
- *RegDeleteValue()*: Borra una entrada de valor.

1.3 Los directorios básicos de Linux

En sistemas tipo Unix, como Linux, no hay un único almacén central de información de configuración. Esta información se muestra en ficheros dentro del sistema de ficheros. Distintas distribuciones utilizan estructuras ligeramente diferentes, pero en general se suelen tener estos directorios relativos a configuración:

- */etc*: es el almacén principal de ficheros de configuración. Algunos de los elementos más importantes dentro de él son:
  - El fichero *fstab*, que guarda la configuración de los discos y particiones.
  - El directorio *X11*, que guarda configuración del sistema de ventanas.
  - El directorio *network*, que guarda información de interfaces de red.

[1] Antiguamente había dos aplicaciones: *regedit.exe*, que se utilizaba en los sistemas operativos descendientes de DOS, y *regedt32.exe*, que se utilizaba en los sistemas operativos descendientes de Windows NT. A partir de Windows XP se fundieron en una sola aplicación que puede ser invocada con cualquiera de los dos nombres.

    - Los directorios *init.d* y *rc*n.*d*, que contienen los *scripts* que se ejecutan al iniciarse el sistema.
- */proc*: sistema de archivos virtual que da información del núcleo y de los procesos. También es posible configurar aspectos del núcleo escribiendo valores en él.
- */sys*: sistema de archivos virtual que da información sobre los dispositivos y sus controladores, además de permitir configurarlos.
- */boot*: ficheros necesarios durante el arranque. Por ejemplo, el gestor de arranque *grub* guarda aquí la configuración del menú de arranque.
- */dev*: ficheros virtuales de dispositivos.
- */home*: carpetas personales de cada usuario. En muchas ocasiones la configuración de las aplicaciones se guarda como archivos ocultos (que empiezan por punto) en la raíz de su carpeta de usuario.

Por otra parte, los entornos de escritorio como GNOME o KDE suelen tener un centro de control con herramientas gráficas que facilitan la modificación del sistema sin tener que conocer la sintaxis y la localización exacta de los ficheros de configuración.

2. Controladores de dispositivo

Los controladores de dispositivo (*device drivers* o, en muchas ocasiones, *drivers* sin más) son programas que permiten a programas de más alto nivel interactuar con un dispositivo sin conocer muchos detalles del dispositivo. De esta manera se simplifica la programación de aplicaciones y, además, se facilita que funcionen con una gran variedad de dispositivos.

Los controladores de dispositivo se pueden ver como una extensión del núcleo del sistema operativo y, en consecuencia, son muy dependientes de éste. En muchos casos, se ejecutan en el mismo nivel de privilegios que el núcleo, lo que hace que un fallo en un controlador de dispositivo pueda hacer fallar a todo el sistema. Por lo tanto, son piezas de software especialmente críticas.

En los siguientes puntos se analiza el funcionamiento de los controladores de dispositivos en Windows y en Linux.

2.1 Controladores de dispositivo para Windows

2.1.1 Tipos de controladores de dispositivo

Windows, a lo largo de sus diferentes versiones, ha ido utilizando distintos modelos de controladores de dispositivos.

En Windows 95 se utilizaban los controladores VxD. Por su parte, Windows NT 4 utilizaba un modelo de controladores propio e incompatible con los utilizados Windows 95.

Windows 98 y Windows 2000 pasaron a utilizar controladores WDM (inicialmente significaba *Windows Driver Model* pero ahora significa *Win32 Driver Model*). La idea era que los controladores fuesen compatibles entre

ambas versiones, de tal manera que no hubiera que crear controladores distintos para las versiones de Windows para el hogar y las profesionales, aunque en muchas ocasiones se siguieron utilizando controladores diferentes. Los controladores WDM fueron pensados para ser compatibles hacia delante pero no hacia atrás, es decir, un controlador para Windows 2000 suele funcionar en Windows XP pero uno de Windows XP no suele funcionar en Windows 2000.

La creación de controladores WDM es muy compleja. Incluso para desarrollar un controlador muy sencillo se deben realizar muchas tareas (gestión del Plug and Play, gestión de la energía, etc.) que requieren la creación de muchas funciones. Por ello, coincidiendo el lanzamiento de Windows Vista, Microsoft lanzó *Windows Driver Foundation* (WDF), que es un entorno (*framework*) para el desarrollo de controladores. Incluye dos modelos de desarrollo:

- El entorno de controladores en modo núcleo (*Kernel-Mode Driver Framework*, KMDF): Utiliza una interfaz en C. Es necesario cuando el controlador necesita manejar interrupciones, acceder a DMA o requiere recursos del núcleo como memoria no paginada.
- El entorno de controladores en modo usuario (*User-Mode Driver Framework*, UMDF): Utiliza un interfaz en C++ y realiza la comunicación entre objetos mediante COM (*Common Object Model*). Son mucho más sencillos y no comprometen la estabilidad del sistema tanto como los KMDF.

Windows Vista también ha introducido un nuevo modelo para los controladores de vídeo, el *Windows Display Driver Model* (WDDM). Permite funcionalidades avanzadas para pintar las aplicaciones y el escritorio utilizando efectos 3D.

La arquitectura de controladores de Windows está formada por una serie de capas. Cuando se hace una petición a un dispositivo, la petición se traduce en una serie de llamadas a funciones de más bajo nivel cada vez, hasta que finalmente se realiza la comunicación con el dispositivo a través de los mecanismos de E/S del procesador.

Desde el punto de vista más alto, se distinguen dos tipos de controladores:

- **En modo usuario.** A su vez, pueden ser de dos tipos:
  - **Controladores de dispositivo virtual (*Virtual Device Driver*, VDD):** Son controladores utilizados para emular aplicaciones de MS-DOS.
  - **Controladores de impresora:** Traducen las operaciones gráficas independientes del dispositivo a órdenes específicas de la impresora. Más adelante se enviarán a un controlador específico en modo núcleo de la impresora instalada.
- **En modo núcleo.** Pueden dividirse en las siguientes categorías generales:
  - **Controladores de sistemas de archivos.** Reciben peticiones de lecturas de archivos y las traducen a peticiones para los controladores del dispositivo correspondiente. Por lo tanto, siempre dependen del soporte de otros controladores de más bajo nivel.

- **Para dispositivos heredados.** Estos controladores se utilizan para dispositivos antiguos, controladores heredados de Windows NT, que no soportan funciones avanzadas como PnP (Plug and Play) o gestión de la energía.
- **Para pantalla.** Se encargan de recibir peticiones gráficas independientes del dispositivo en peticiones específicas para un dispositivo.
- **Controladores WDM.** Se dividen en:
  - **Controladores de función.** Controlan periféricos específicos. Se suelen dividir en dos capas: clase (con la funcionalidad común a un conjunto de dispositivos, como ratones, teclados o discos) y miniclase (con la funcionalidad específica de un modelo concreto).
  - **Controladores de filtrado.** Modifican el funcionamiento de los controladores de función y pueden ejecutarse antes o después del controlador de función al que complementan.
  - **Controladores de bus.** Controlan un bus como PCI, USB, etc. Se encargan de labores como detectar los dispositivos conectados al bus y controlar la gestión de energía del bus.

2.1.2 Instalación

El objetivo de la instalación es guardar en el registro del sistema toda la información que necesita Windows para poder utilizar el controlador. Esta información se utiliza durante la carga de los controladores para generar el árbol de dispositivos del sistema.

2.1.2.1 Ficheros

Ficheros necesarios para la instalación:

- Fichero imagen del controlador (Fichero SYS): Contiene el controlador. Cuando se instala el controlador, se copia este fichero al directorio *%windir%\system32\drivers*. Posteriormente, cuando Windows necesita cargar el controlador para utilizar un dispositivo, lo lee de este directorio.
- Fichero de información de configuración del dispositivo (Fichero INF): Contiene la información sobre cómo se instala el dispositivo y qué controladores se pueden utilizar para controlarlo. Los ficheros INF de todos los controladores instalados en el sistema se guardan en *%windir%\inf*. Cuando se detecta un nuevo dispositivo, primero se busca en este directorio por si hay ya algún controlador instalado que sea compatible con él.

2.1.2.2 Identificadores

A cada dispositivo físico del sistema se le asigna un identificador único, llamado *Device Instance ID* (DIID), que permite distinguirlo de todos los demás. Durante la instalación del dispositivo se asocia el DIID con un controlador, de forma que cuando el sistema detecta la presencia de ese dispositivo sabe qué controlador tiene que cargar. Esta asociación se guarda en el registro.

El gestor de PnP genera los identificadores (DIID) durante la enumeración de los dispositivos a partir de información que se encuentra almacenada en el hardware del dispositivo. Concretamente, el DIID de un dispositivo se genera combinando estos dos identificadores:

- ID de dispositivo (*Device ID*, *DID):* Identifica a un modelo de componente hardware en concreto. El formato de este identificador depende del bus al que está conectado el dispositivo.
- ID de instancia (*Instance ID*, *IID):* Es un identificador que permite distinguir entre varias instancias del mismo dispositivo hardware. Generalmente, este identificador depende de la posición del dispositivo dentro del bus.

Construyendo el DIID de esta forma (DIID = DID\IID) se tiene la seguridad de que nunca habrá dos dispositivos con el mismo DIID en el sistema.

Además del DID, cada componente hardware tiene almacenado un conjunto de identificadores adicionales que se utilizan durante el proceso de instalación para decidir qué controlador es el más adecuado para controlar ese dispositivo. El sistema operativo trata siempre de encontrar el controlador que mejor se ajusta a las características de ese dispositivo utilizando estos identificadores:

- *Hardware ID (HID):* Cada componente hardware almacena una lista de Hardware IDs que indican de qué dispositivo se trata. Esta lista está ordenada desde el identificador más específico al más general. El primer ID de esta lista es el Device ID (DID). Cuando se instala un dispositivo, si el sistema operativo no encuentra un controlador específico para el DID, se empiezan a buscar controladores para los HID del componente hasta que se encuentre un controlador que sirva para ese dispositivo. Los HID para los que es compatible un controlador vienen especificados en su fichero INF.
- *Compatible ID (CID):* Cada componente hardware almacena también una lista de IDs de componentes compatibles por si no se encuentran controladores para ninguno de los HID del dispositivo. La lista está también ordenada del más adecuado al menos adecuado.

2.1.2.3 Proceso de instalación

La instalación de un controlador se puede iniciar de forma manual seleccionando la opción "Instalar" del menú contextual de su fichero INF (pulsando el botón derecho sobre ese fichero). El sistema operativo procesa el fichero INF e instala el controlador añadiendo la información necesaria al registro y copiando los ficheros del controlador a los directorios *%windir%\inf* y *%windir%\system32\drivers*. Este tipo de instalación se utiliza cuando el controlador no es Plug and Play o cuando el controlador no se utiliza para controlar un dispositivo físico.

El gestor de Plug and Play inicia la instalación de un controlador de forma automática cuando el sistema detecta la presencia de un nuevo dispositivo hardware. En primer lugar, el bus envía al gestor de PnP el DIID que identifica al nuevo dispositivo. A continuación, el gestor PnP busca en el registro por si hay ya algún controlador instalado para ese DIID. Si no lo hay, se intenta

instalar un controlador para ese DIID sin la intervención del usuario. Se busca en el directorio *%windir%\inf* el fichero INF de algún controlador que esté instalado en el sistema, que esté asociado a uno de los HID del dispositivo y que además esté firmado digitalmente. Si se encuentran varios controladores posibles, se elige el más adecuado siguiendo ciertas reglas, se procesa el fichero INF y se asocia el DIID del dispositivo a ese controlador, para que el sistema sepa qué controlador tiene que utilizar la próxima vez que se detecte el dispositivo.

Generalmente, cuando se conecta hardware nuevo, el sistema operativo no encontrará un fichero INF compatible con un HID del dispositivo o necesitará información adicional que no se encuentra en el fichero INF, por lo que necesitará la colaboración de un usuario (con privilegios de administrador) para completar la instalación. El gestor de PnP activa el Asistente de Instalación de Hardware para solicitar al usuario la ubicación de un fichero INF que sea compatible con alguno de los identificadores (HID o CID) del dispositivo.

Tras el proceso de instalación, se guarda en el Registro información del controlador instalado en la clave KHLM\SYSTEM\CurrentControlSet\Services. y del dispositivo en HKLM\SYSTEM\CurrentControlSet\Enum.

2.1.2.4 Firmas digitales y catálogos

En Windows se utiliza un sistema de control de calidad para comprobar la autenticidad y validez de los controladores que se instalan y verificar que funcionan correctamente. Este mecanismo se basa en la utilización de firmas digitales. Cada fichero de un paquete de instalación se firma digitalmente. Cuando se instala el controlador se comprueba su firma para asegurarse de que el controlador ha pasado los controles de calidad de Microsoft y que los ficheros no han sido modificados. Si el controlador no está firmado o la firma no es válida, se informa al usuario de que no es aconsejable instalar dicho controlador porque no ha sido verificado.

Los ficheros de catálogo (.cat) contienen las firmas digitales de todos los ficheros que se distribuyen en el paquete de instalación de un controlador y permite verificar la validez de los mismos. Cada firma es un checksum criptográfico del contenido del fichero que se genera utilizando una clave privada de Microsoft.

Las firmas digitales y los catálogos son generados por el WHQL de Microsoft. (*Windows Hardware Quality Laboratory*). Cuando un fabricante desarrolla un controlador, lo envía a este laboratorio para que los técnicos de Microsoft lo prueben y, si pasa los criterios de calidad, lo firmen digitalmente mediante un fichero de catálogo. El controlador firmado se devuelve al fabricante para que lo pueda distribuir.

WHQL sólo firma los controladores que pertenecen a una de las clases de dispositivo definidas por Microsoft. Si un dispositivo no pertenece a ninguna de estas clases, todavía se puede firmar digitalmente utilizando la tecnología Microsoft Authenticode. En este caso, el catálogo viene firmado por el fabricante en lugar de por Microsoft.

Se puede elegir qué hacer cuando un controlador no viene firmado digitalmente. Las opciones disponibles son:

- *Ignorar*: Se instala el controlador de todas formas sin avisar al usuario de que no está firmado.
- *Avisar*: Se avisa al usuario de que el controlador no está firmado para que decida qué hacer.
- *Bloquear*: No se permite instalar controladores que no estén firmados.

La versión de 64 bits de Windows Vista no permite instalar controladores en modo núcleo que no estén firmados.

2.1.2.5 Coinstaladores, instaladores de clase y aplicaciones de instalación

Muchas veces la instalación de un controlador requiere un conjunto de tareas adicionales que no están contempladas en el mecanismo básico de instalación. Windows permite ampliar el mecanismo de instalación básico utilizando coinstaladores, instaladores de clase y aplicaciones de instalación. Estos tres componentes se comunican con el mecanismo de instalación básico mediante una librería llamada *SetupAPI*.

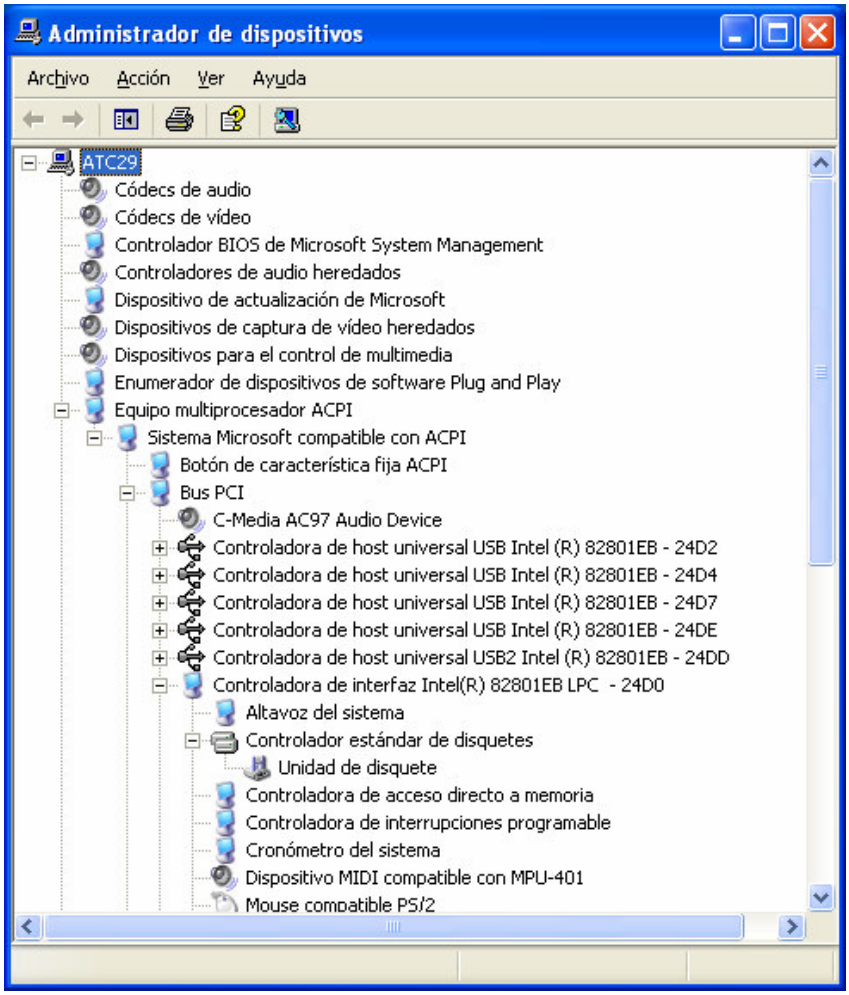
- *Coinstaladores:* Un coinstalador es una DLL que se utiliza durante el proceso de instalación para realizar operaciones de instalación específicas que no están incluidas en la instalación básica.
- *Instaladores de clase:* Un instalador de clase permite añadir una nueva clase de dispositivos a la lista de clases predefinida de Windows.
- *Aplicaciones de instalación (Instaladores):* Estas aplicaciones presentan la interfaz de instalación del controlador y dirigen todo el mecanismo de instalación mediante las funciones de la librería *SetupAPI*.

2.1.3 Carga de controladores

Los controladores en Windows se pueden cargar de dos formas:

- Mediante carga explícita. Se cargan los dispositivos presentes en HKLM\SYSTEM\CurrentControlSet.
- Mediante enumeración. Ocurre cuando el subsistema de gestión de PnP carga los controladores de los dispositivos que controlador de bus encuentra durante el proceso de enumeración.

El gestor de PnP realiza un proceso de listado de los dispositivos conectados a los distintos buses para formar un árbol de dispositivos. Este árbol se puede mostrar en el Administrador de Dispositivos mediante la opción *Ver/Dispositivos por conexión*. La siguiente figura muestra un ejemplo:



2.2 Controladores de dispositivo para Linux

2.2.1 Tipos de controladores de dispositivo

El desarrollo, la utilización y la instalación de controladores de dispositivo en Linux es muy diferente a la de Windows, en gran parte por dos motivos: por un lado, la filosofía de software libre hace que se prefiera un controlador del que se tenga el código fuente y que se pueda redistribuir con licencia GPL; por otro, al no ser un sistema con una implantación masiva en el escritorio, no recibe la misma atención por parte de los fabricantes de hardware. Esto último significa que muchos fabricantes no proporcionan controladores de sus dispositivos para Linux. Esto sería un problema menor si no fuese porque muchas veces tampoco proporcionan especificaciones que permitan a personas ajenas al fabricante construir los controladores. Ante esta situación, a los programadores de Linux sólo les queda recurrir a la ingeniería inversa para intentar descubrir cómo funciona el dispositivo, lo que no siempre es factible y, en los casos en los que se consigue, en muchas ocasiones es con un rendimiento inferior al que tendría si estuviera desarrollado conociendo las especificaciones y las características del hardware como lo conoce el fabricante del dispositivo.

2.2.2 Modelo de dispositivos de Linux

Con la versión 2.6 del núcleo se creó un nuevo modelo de gestionar los dispositivos hardware, denominado *Linux Device Model*. Entre sus principales características están:

- Estructuración en buses (dispositivos que sirven de conexión a otros dispositivos), dispositivos y controladores. Las implementaciones de un bus pueden, por ejemplo, buscar controladores de un dispositivo que se conecta a ese bus.
- Clases: hay distintos tipos de dispositivos que tienen características en común. Ejemplos de clases son discos, puertos serie, etc. Permite buscar un tipo de dispositivo independientemente de a qué bus se conecte.
- Subsistemas: Visión jerárquica de la estructura del sistema. Por ejemplo, *devices* da una visión jerárquica de todos los dispositivos, *bus* la da desde el punto de vista de los buses, *class* la da por clases de dispositivo, etc.
- hotplug: permite detectar dispositivos *en caliente*, es decir, conectados después de haber arrancado el sistema.
- coldplug: permite detectar dispositivos conectados antes de haber arrancado el sistema.
- udev: es el gestor de dispositivos. Se encarga de manejar los nodos en */dev*. En sistemas Unix tradicionalmente se mostraban en */dev* todos los posibles dispositivos que podía manejar un sistema; udev hace que se muestren sólo los realmente instalados. Funciona como un demonio que recibe eventos del núcleo cuando se detecta un nuevo dispositivo. Analiza una serie de reglas para dar nombre al dispositivo y ejecutar los *scripts* que sean necesarios para configurarlo.
- kobjects (*kernel objects*): es una estructura que permite implementar conceptos de orientación a objetos en el núcleo, que está escrito en C. Un kobject incluye un nombre, cuenta de referencias, padre y tipo. Cuando una estructura incluye un kobject está formando parte de la jerarquía de objetos del núcleo.
- sysfs (*system filesystem*): muestra información de los dispositivos como un sistema de ficheros virtual en */sys*.
- HAL (*Hardware Abstraction Layer*): es un demonio que, uniendo hotplug, sysfs y udev, proporciona una visión uniforme de los dispositivos del sistema y permite acceder a ellos a través de un interfaz estándar. El demonio mantiene una lista de dispositivos con un conjunto de parejas clave-valor que dan información sobre él.

2.2.3 Instalación y carga

Cuando se dispone del código fuente, se puede optar por compilar el código del controlador al mismo tiempo que el propio núcleo del sistema operativo e integrarlo en la misma imagen. Pero, por otra parte, no interesa que todos los controladores posibles formen parte de la imagen del núcleo: la mayor parte de

los controladores no serán necesarios porque nunca se conectará un dispositivo que lo necesite al sistema, y otros serán necesarios sólo temporalmente, como los dispositivos que se conectan por USB en ocasiones. Además es necesario un mecanismo para utilizar controladores de los que no se dispone el código fuente.

Para poder cargar sólo los controladores que interesen en un momento dado, se utilizan módulos cargables del núcleo. Estos módulos son código binario que el núcleo puede cargar. Están en */lib/modules* y tienen la extensión *.ko*.

Hay que tener en cuenta que, en general, un controlador compilado para una versión del núcleo no funciona con otras versiones. Como hay distribuciones que realizan cambios al núcleo oficial, puede haber también controladores que funcionen con una distribución y no con otra, aunque ambas estén basadas en la misma versión del núcleo.

Los módulos se pueden obtener de varias formas:

- Si están disponibles en el código fuente del núcleo, al compilar el núcleo se puede seleccionar qué módulos compilar.
- Si se distribuye el código fuente, será necesario compilarlo para la versión del núcleo que se tenga.
- Si se dispone sólo del binario, será necesario copiar el archivo correspondiente. Los controladores así distribuidos suelen ir acompañados de un *script* de instalación.
- La propia distribución puede tener en su sistema de gestión de paquetes el controlador y, en ese caso, se puede instalar utilizando el gestor de paquetes correspondiente.

La instalación consiste en copiar el módulo al lugar correspondiente y hacer que se cargue. Hay que tener en cuenta que puede ser necesario además un proceso de configuración y activar *scripts* adicionales, siendo este proceso muy dependiente del controlador a instalar.

Durante el arranque del sistema se cargan los módulos que se encuentren listados en */etc/modules*. Para cargar un módulo posteriormente, se puede utilizar la utilidad *modprobe*. Para ver si un módulo está cargado, se puede utilizar *lsmod*.

3. Información del sistema

Hay muchas herramientas para conocer información del sistema. Ejemplos para Windows son SiSoft Sandra, Lavalys Everest (antes llamado AIDA32), FreshDiagnose o CPU-Z. En Linux, los gestores de ventanas suelen tener su propia aplicación (KInfoCenter o hardinfo), además de existir múltiples herramientas de línea de comandos, como dmidecode.

A veces interesa obtener información del sistema desde un programa. En Windows, se puede recurrir a las funciones de información del sistema del API. Cuando estas funciones no dan información suficiente, se puede recurrir a las de acceso al Registro para acceder a la clave HKLM\HARDWARE, que se crea cada vez que se arranca el sistema con información del hardware detectado. En Linux

se pueden utilizar funciones del API POSIX o leer del sistema de archivos virtual /sys.

En entornos empresariales, en los que es necesario controlar una gran cantidad de máquinas, son necesarias herramientas más avanzadas. Se han desarrollado diversos estándares para la gestión de entornos distribuidos. Entre los más importantes destaca WBEM (Web Based Enterprise Management, Sistema de Gestión Empresarial basado en Web). Sus elementos más importantes son:

- Un modelo de información del sistema denominado CIM (*Common Information Model*) que, mediante una jerarquía de objetos, permite describir el conjunto de elementos de un sistema.
- Un sistema de acceso a los datos basado en protocolos abiertos. De esta forma se pueden desarrollar herramientas clientes que interactúen con el subsistema WBEM a través de HTTP o de *web services*, por ejemplo.

WBEM puede recoger información de diversos elementos de más bajo nivel, como por ejemplo:

- **Windows Management Interface (WMI).** Es un subsistema de Windows que permite obtener información del sistema. Los controladores de dispositivo WDM se comunican con este subsistema para dar información y recoger valores de configuración.
- **Desktop Management Interface (DMI).** Es un estándar que permite que la BIOS exporte información sobre el sistema de forma predefinida que luego pueda ser tratada por otros programas.
- **Simple Network Management Protocol (SNMP).** Es un protocolo de la capa de aplicación de los protocolos de Internet para gestionar elementos conectados a una red.