

El Registro de Windows y temas relacionados

Índice

| | | |
|---------|--|----|
| 1. | Introducción al Registro..... | 4 |
| 1.1. | Puntualización | 4 |
| 1.1. | ¿Cómo se carga el S.O. y sale el escritorio? | 4 |
| 1.2. | A partir de aquí se inicia el escritorio | 4 |
| 1.3. | Los posibles errores en la carga | 4 |
| 1.4. | ¿Pero qué es el Registro? | 5 |
| 1.5. | ¿Qué compone el Registro? | 5 |
| 1.5.1. | User.dat..... | 6 |
| 1.5.2. | System.dat | 6 |
| 1.5.3. | Policy.pol..... | 6 |
| 2. | Unas pequeñas notas sobre WinME | 7 |
| 3. | Config.sys..... | 7 |
| 3.1. | Parámetros básicos | 7 |
| 3.2. | Otros parámetros del Config.sys | 9 |
| 3.2.1. | ACCDATE=disco1+ - disco2+ - | 9 |
| 3.2.2. | BREAK [ON OFF]..... | 9 |
| 3.2.3. | BUFFERS=n[,m] ; BUFFERSHIGH=n[,m] | 10 |
| 3.2.4. | DEVICE= ; DEVICEHIGH=..... | 10 |
| 3.2.5. | DOS=HIGH LOW[,UMB ,NOUMB][,AUTO ,NOAUTO] | 11 |
| 3.2.6. | DRVPARAM= | 11 |
| 3.2.7. | FCBS=x ; FCBSHIGH=x | 11 |
| 3.2.8. | FILES=nn ; FILESHIGH=nn | 11 |
| 3.2.9. | INSTALL= ; INSTALHIGH= | 12 |
| 3.2.10. | LASTDRIVE= ; LASTDRIVEHIGH= | 12 |
| 3.2.11. | NUMLOCK=[ON OFF]..... | 12 |
| 3.2.12. | REM | 12 |
| 3.2.13. | SET variable=xxxxxxxx | 12 |
| 3.2.14. | SHELL=[[disco:]path]programa [parametros] | 12 |
| 3.2.15. | STACKS= ; STACKSHIGH=..... | 12 |
| 3.2.16. | SWITCHES= /F /K /N /E[:n]..... | 13 |
| 4. | AutoExec.Bat..... | 13 |
| 4.1. | Parámetros de configuración | 13 |
| 4.2. | Condiciones de entorno | 14 |
| 4.3. | Empieza la optimización de nuestro sistema. | 15 |
| 4.4. | Confusión existente en torno al MSCDEX | 15 |
| 5. | ¿Cómo utiliza el Registro Windows98? | 17 |
| 5.1. | Las herramientas | 17 |
| 5.1.1. | Un comprobador: Scanreg y Scanregw..... | 17 |
| 5.1.2. | Un editor: Regedit. | 18 |

| | | |
|-------|--|----|
| 6. | Estructura y tipos de datos del registro | 18 |
| 6.1. | Las claves..... | 18 |
| 6.2. | Los valores | 18 |
| 6.3. | Las raíces | 18 |
| 7. | Las claves del registro..... | 19 |
| 7.1. | HKEY_CLASSES_ROOT | 19 |
| 7.2. | HKEY_CURRENT_USER..... | 20 |
| 7.3. | HKEY_LOCAL_MACHINE | 20 |
| 7.4. | KKEY_USERS | 21 |
| 7.5. | HKEY_CURRENT_CONFIG..... | 22 |
| 7.6. | HKEY_DYN_DATA | 22 |
| 8. | El Registro de Windows 98 comparado con Windows 95 y NT/2000..... | 22 |
| 9. | Métodos de introducir claves en el registro (archivos REG e INF) | 22 |
| 9.1. | Archivo .Reg | 22 |
| 9.2. | Archivo .Inf..... | 23 |
| 10. | El Registro y el P&P | 23 |
| 11. | Otros valores importantes del registro..... | 25 |
| 11.1. | La lista MRU | 25 |
| 11.2. | Los modems | 26 |
| 11.3. | TCP/IP de la subclave MSTCP | 26 |
| 12. | Msdos.Sys - Parámetros de configuración | 27 |
| 12.1. | [Paths]..... | 27 |
| 12.2. | [Options] | 27 |
| 13. | Vigilancia del sistema (SFC) | 30 |
| 14. | Anexo 1: Modos | 31 |
| 14.1. | Modo Real Y Modo Protegido..... | 32 |
| 14.2. | Control de procesos. Protección..... | 32 |
| 14.3. | Memoria Virtual. Su mecanismo. | 33 |
| 14.4. | Virtualización del hardware..... | 34 |
| 14.5. | Modo virtual 8086 | 35 |
| 14.6. | Multitarea real | 35 |
| 15. | Anexo 2: El BUG de Windows 98 y SE | 37 |
| | Agradecimientos | 39 |
| | Bibliografía..... | 39 |

Por Chiquito de la Calzada
chiquito_de_la_calzada@hotmail.com

El disponer de buena información siempre ayuda en casos de apuro. A los apasionados que revisan todos los recovecos de los sistemas operativos Windows © MS.

1. Introducción al Registro

1.1. *Puntualización*

El registro que utiliza MS en sus windows es de dos tipos reg4 y reg5, el primero utilizado en los núcleos w9x y el segundo en win2k. Cuando se utilizan las herramientas del registro muchas veces se ignora que el programa regedit no es el mismo que el regedt32 y que cada uno se corresponde a cada uno de los tipos de registro mencionado. Y aunque parece que ambos funcionan bien en w2k(reg5) yo no estaría seguro. Por otra parte todo lo que vamos a ver a partir de ahora es tipo reg4, editable con regedit.exe y para win98 especialmente. Ya echaremos un vistazo a estas herramientas de edición del registro, e incluso a editar scripts o inis para introducir datos en el mismo de forma sencilla.

1.1. *¿Cómo se carga el S.O. y sale el escritorio?*

El proceso de arranque es una buena fuente de información para ayudarnos a resolver los problemas del registro y del s.o..

- 1) Se carga el registro, que contiene la información básica.
- 2) Se lee el system.ini, para la configuración heredada que pueda existir.
- 3) Se carga el kernel (núcleo) por medio de la librería dinámicakernel32.dll
- 4) La interfaz gráfica, los Gdi.exe y Gdi32.dll.
- 5) El código referente a los usuarios, User.exe y User32.exe.
- 6) Las fuentes y recursos.
- 7) El Win.ini, configuración de usuario y programa heredado.

1.2. *A partir de aquí se inicia el escritorio*

La interfaz de órdenes y directivas a nivel de máquina.

Los componentes del escritorio.

En caso de estar en red, el inicio de sesión. Una vez iniciada, las órdenes de inicio de sesión y se aplican las directivas a nivel de usuario.

Si se inicia sesión se cargará los archivos de directivas específicas de usuario, si no hay Inicio de sesión se aplicarán los valores predeterminados. Y si esta es en Red el usuario será conectado a la misma.

1.3. *Los posibles errores en la carga*

- **Error en el registro y que se reiniciará.**
Windows debería reiniciar en modo MS-DOS y ejecutar Scanreg.exe para corregir el problema.
- **Memoria insuficiente.**

Podría incluso informar de la forma de solucionar el problema. Normalmente reiniciar en Sólo símbolo de sistema y ejecutar Scanreg /Fix. Si después de aplicar el scanreg y su parámetro /Fix, se repite el mensaje de falta de memoria, hay que liberar memoria convencional.

Normalmente yo arrancaría con un disquete de inicio simple, con un autoexec.bat y config.sys al mínimo.

- **No se encuentra un archivo o aplicación necesaria para ejecutar windows. Incluso indica el nombre del archivo o aplicación.**

Suele ocurrir cuando un controlador de dispositivo virtual (VxD) al cual hace referencia el System.ini o el registro no está o está corrompido; o que alguno de los valores VxD estáticos del registro contiene datos inválidos, como estar vacío o sólo tiene espacios.

Si se ha eliminado alguna aplicación se reinstala y se vuelve a desinstalar con el procedimiento correcto, es decir, desde el componente Agregar/quitar.. Del panel de control o del archivo desinstalador del propio programa. (Si no aparece en la lista y/o no lleva desinstalador, cosa típica de programas antiguos (sobre todo MS-DOS), hay que eliminarlo manualmente y borrar las referencias al archivo erróneo en System.ini y en el registro. (Ya veremos cómo.)

Si el archivo que da error lleva la extensión 386, con poner un (;) punto y coma delante de la referencia en el System.ini, bastará. Si por el contrario es un VxD, estará en el registro.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD, es la rama que contiene los valores de los VxD.

Se debería Editar el registro y borrar los valores vacíos o sólo con espacios en blanco.

A grosso modo son los típicos errores en el arranque debidos a problemas de registro y/o archivos de compatibilidad.

1.4. *¿Pero qué es el Registro?*

El registro es un simple archivador, en el que residen todos los datos de configuración del sistema. De windows, de hardware, de aplicaciones y de usuarios.

Contiene estos datos en forma jerarquizada. De superior a inferior. Y como lo contiene todo, en teoría sobra cualquier configuración en los archivos de inicio, si esos tan conocidos como Autoexec.bat o Config.sys. Además proporciona los datos dinámicamente.

Su pariente más cercano son los famosos INI, de hecho cada clave del registro es como uno de estos archivos, salvando cierta distancia.

En realidad, windows permite los archivos INI y los autoexec.bat y config.sys por mera compatibilidad hacia atrás. Si todo lo que tenemos son aplicaciones Win32 estos archivos no son nada útiles..

1.5. *¿Qué compone el Registro?*

Aunque es de forma lógica como un almacén de datos, en realidad consta de tres archivos distintos para su máxima flexibilidad de configuración de red.

Y en cada uno se almacena información de una categoría.

1.5.1. User.dat

Contiene en forma de perfil, toda la información de usuario. Nombres de inicio de sesión, la configuración del escritorio, el menú de inicio, etc. Cuando se configura cualquiera de estos aspectos, User.dat cambia y se almacena de forma oculta en \Windows, aunque puede estar -en caso de configurar varios perfiles- en \Windows\Profiles y en todo caso si se está en Red en un servidor que sea controlador de dominio (PDC).

1.5.2. System.dat

Es idéntico al anterior, pero esta vez son las configuraciones de hardware, plug&play y las aplicaciones las que se almacenan aquí. Siempre está en el ordenador Local y está oculto en \Windows.

1.5.3. Policy.pol

No es obligado que exista. Normalmente se utiliza por los administradores del sistema o red, para sustituir valores de los dos anteriores. Conocidas como las directivas del sistema, y son datos específicos de la red.

MS le otorga al Registro unas mejoras respecto al modo anterior de archivos ini y los de inicio.

Se considera que se utiliza menos memoria tanto en memoria real como en modo protegido. Por ello se consiguen inicios del sistema más breves y un aumento del rendimiento del sistema.

El soporte que ofrecen en caché es mucho mayor, reduciendo el tiempo en la consulta de sus valores.

Es más fácil detectar posibles corrupciones en los datos y en el funcionamiento. Un ejemplo claro es al apagar mal el sistema, sea por la causa que sea; se obliga a ejecutar el comprobador del registro (scanreg) y por supuesto el famoso scandisk.

Mediante el comprobador del registro se realiza un backup diario, llegando a tener cinco copias del mismo. (Se puede configurar para aumentar y/o disminuir ese número)

Se utiliza un único sitio de datos para el hardware, las aplicaciones, los dispositivos y sus controladores y todos los parámetros y además se puede recuperar en caso de fallo.

Los usuarios y los administradores mediante las herramientas del panel de control y otras, pueden configurar varias opciones del ordenador evitando los errores sintácticos.

Se tienen funciones independientes de la red para la consulta remota de datos.

Tiene un límite superior a 64Kb, por lo que se aumenta la posibilidad de instalación de aplicaciones con librerías dinámicas o dll's.

Y como la información de un usuario se puede contener en un servidor, facilita el acceso a preferencias de escritorio y acceso a la red por parte de los usuarios desde cualquier ordenador en que se conecten. Y además se puede forzar la aplicación de directivas a usuarios, grupos o a todos.

2. Unas pequeñas notas sobre WinME

Si ya en Windows 98 el archivo autoexec.bat empieza a ser innecesario, en WinME carece de funciones, dejando sólo su existencia para cierta compatibilidad.

Veamos que hace WinME con este archivo:

Actualmente, en Windows ME, el autoexec.bat no se ejecuta (únicamente, está en modo documental, y únicamente por motivos de compatibilidad.)

Solo tienen sentido allí los comandos SET y PATH, es decir los comandos que establecen las variables de entorno. Aún así, el autoexec no se ejecuta. Actualmente las variables de entorno (las que producía el comando SET) y el PATH (camino de búsqueda para ejecución de programas), están en el registro. Se pueden modificar mediante el comando MSCONFIG.

¿qué sentido tiene entonces el autoexec?... veamos su funcionamiento. Las variables reales de entorno y el PATH están en el registro. Pero Windows ME lo que hace es 'sincronizar' con el autoexec. Es decir, lo que modifiquemos en el registro lo grabará en el autoexec.

Además se graba en el registro un 'check' de control del tamaño y fecha del autoexec. Cada vez que arranquemos o finalicemos Windows, verifica es 'check'. Si es el mismo no hace nada.

Si es diferente lee su fecha. Si la fecha es menor de la que está almacenada en el registro, considera un autoexec inválido, lo borra y vuelve a generarlo con el contenido de las variables del registro.

Si la fecha es mayor, supone que algún programa ha modificado su contenido. Lo 'lee' y el contenido de las variables SET y PATH lo reincorpora al registro eliminado las líneas que no corresponden a este contenido.

Esto está realizado así por motivos de compatibilidad de viejos programas que incorporan su 'path' y variables de entorno en el autoexec. De esta manera Windows ME se sincroniza con el autoexec.

3. Config.sys

3.1. *Parámetros básicos*

Partiendo de la base clara que el archivo config.sys ha dejado de ser necesario, tan sólo queda como un recuerdo al MS-DOS. Aunque para las configuraciones de países distintos a USA se añaden unas líneas para las configuraciones regionales.

Para España podrían quedar:

```
device=C:\WINDOWS\COMMAND\display.sys con=(ega,,1)
```

```
Country=034,850,C:\WINDOWS\COMMAND\country.sys
```

No se necesitan más líneas.

El Windows 3.1 fue la primera interfaz gráfica sobre MS-DOS que fue capaz de superar la barrera de 1Mb. Y para ello utilizaba el procesador en modo protegido.

MS creó el controlador necesario, DPMI (Dos Protected Mode Interface), y Que conocemos como el Himem.sys.

Como siempre la necesidad de compatibilidad hace que windows98 utilice la misma interfaz. Aunque siempre es opcional el habilitarlo desde el config.sys, Windows98 lo cargará de todas formas.

Además cargará algunos drivers para dejar una zona de memoria baja para los dispositivos SCSI que necesiten transferir datos en DMA.

Cualesquiera otros drivers de dispositivos son totalmente innecesarios en el config.sys, es más puede causar mal funcionamiento del propio S.O. Así que lo normal es revisarlo después de la instalación de programas, pues muchos todavía incorporan líneas al mismo, cuando no son necesarias.

El comando DEVICE o DEVICEHIGH es el encargado de especificar los drivers de dispositivos en config.sys.

Por supuesto hay algunos que nos pueden ser de utilidad en casos concretos, como correr programas DOS antiguos o juegos que necesitan utilizar memoria en modo MSDOS. Concretamente se llama EMM386.exe.

Su funcionamiento se basa en la memoria real, que está por debajo del mega, en concreto 640Kb.

Desde esta cantidad al mega, existen 196Kb y de ellos 32 son utilizados por la bios de la tarjeta gráfica.

En cierto momento, surgió la idea de ocupar ese espacio para meter las propias rutinas de MS-DOS y así dejar libre espacio en los primeros 640Kb.

La primera versión que empleo esta idea fue la MS-DOS 5.0. Y de hecho windows98 se identifica con MS-DOS 7.10.

Así pues, MS-DOS 5.0 modificó el código y cargaba todo o parte de sus rutinas en ese espacio, y con el EMM386 se encargó de recuperarlo y utilizarlo para residentes en la memoria.

Se le denominó UMB /Upper Memory Block/.

Pero aún así, queda espacio libre hasta llegar al mega.

También aparecieron dos controladores, como XMS (Extended Memory System) para memoria extendida; y la EMS (Expanded Memory System), ésta última utilizaba una ventana apuntando a bloques de 64Kb en cualquier posición por encima del mega. De esta forma, cambiando las direcciones, se utilizaban páginas de 64Kb superior al mega.

El subsiguiente cambió del procesador 8086 y posteriores, sirvió para manejar la memoria en modo protegido por encima del mega. Y se modificó al controlador EMM386 para admitir esa funcionalidad.

Resumiendo: EMM386 tiene (o puede tener) tres funciones básicas:

- 1) Control de la memoria EMS (LIM) y creación del marco de pagina
- 2) Soporte a la memoria UMB.
- 3) Soporte a la memoria HMA (HIGH).

Es importante indicar, que si ponemos el driver EMM386.EXE en el config.sys, entonces *SI* que es obligatorio poner primero el HIMEM.SYS en dicho archivo.

Además, también es importante resaltar, que aunque pongamos el parámetro RAM, esto solo indica que esta memoria estará disponible para programas que "sepan"

utilizarla. En particular si queremos que el propio MsDOS pueda utilizarla, se deberá incorporar en el config.sys (en cualquier sitio), la línea:

DOS=UMB

Y por ultimo, también, para que el propio MsDOS pueda utilizar la memoria HIGH (HMA), se deberá igualmente incorporar la línea:

DOS=HIGH

Ambas líneas, pueden resumirse en: DOS=HIGH,UMB (pero recordar que solo tendrán sentido si tenemos el EMM386 con el parámetro RAM, y esto además obliga a tener el HIMEM.SYS)

3.2. Otros parámetros del Config.sys

Lo primero y mas importante, es recordar que NO hace falta ningún parámetro en el config.sys (excepto el DOS=HIGH, RAM visto).

Igualmente quiero insistir en tres cosas:

- 1) Estos parámetros únicamente tendrán efecto en MsDOS no en Windows.
- 2) Cuando me refiero a MsDOS, me estoy refiriendo a MsDOS "puro". Las ventanas MsDOS bajo Windows no se ven afectadas por estos parámetros. Incluso la apertura de ficheros se hace a través de Windows en estos casos.
- 3) Lo que he comentado en los puntos 1) y 2), es la teoría. La práctica nos demuestra, que efectivamente, si que influyen en Windows: pero NEGATIVAMENTE. Es decir, jugando con estos parámetros, lo único que podemos conseguir es que Windows funcione peor.

Por curiosidad vamos a ver estos parámetros y comentar su efecto tanto en MsDOS como en Windows.

NOTA: Algunos de los parámetros que veremos a continuación, pueden terminar en la palabra HIGH. Esto indicará que el MsDOS lo cargará en memoria alta si tenemos activo el EMM386.

Pero esto no es necesario (y además es contraproducente), si además hemos especificado el parámetro DOS=HIGH,UMB que vimos anteriormente, el propio MsDOS ya cargará automáticamente en memoria alta y sin necesidad de "forzar" esta situación.

3.2.1. ACCDATE=disco1+|- disco2+|-

Nos guardará la fecha de la ultima vez que hemos accedido a los archivos de un disco o nó (en función del + o del -). Este parámetro no tiene efecto en Windows, ya que por defecto Windows nos informa de estos accesos.

3.2.2. BREAK [ON|OFF]

Sirve para activar la manera de "parar" un programa MsDOS. En un programa MsDOS, pulsando CTRL-C podemos casi siempre teclear CTRL-C y pararlo. Pero el MsDOS, únicamente chequea el que hayamos pulsado CTRL-C, cuando va a escribir en pantalla o cuando va a leer desde teclado. Si nuestro programa no está haciendo ninguna de estas dos cosas en ese momento, no se parará. Si tenemos BREAK ON en el config.sys, el MsDOS también chequeará el que pulsemos CTRL-C cuando va a leer o escribir en

disco. De esta manera, aunque nuestro programa no escriba en pantalla, se supone que al menos, accederá al disco y por tanto de esta manera podremos pararlo.

No tiene ningún efecto en Windows, y este parámetro no será pernicioso tampoco para Windows. Es conveniente tenerlo.

NOTA: En cualquier ventana MsDOS, podemos teclear BREAK y veremos la situación en la que estamos.

3.2.3. **BUFFERS=n[,m] ; BUFFERSHIGH=n[,m]**

Asigna memoria "real" para contener bufferes de disco. Recordar que cada sector de disco son 512 bytes, por tanto cada buffer que asignemos, nos robará medio k de la memoria real.

No tiene ningún efecto bajo Windows ni bajo las ventanas MsDOS bajo Windows.

Únicamente tiene efecto en MsDOS puro. Y también tiene efecto al "arrancar" Windows, antes de que este entre en modo protegido. Antes de que entre el procesador en ese modo, se está utilizando todavía el antiguo sistema de apertura y carga de ficheros en MsDOS. Por tanto se ahorraría "algo" de tiempo al arrancar Windows, si este parámetro fuese elevado.

Mi consejo es NO ponerlo. Solo sirve para ahorrar unos segundos en el arranque de Windows. Por contra, nos disminuirá muchísimo nuestra memoria real. Recordar que esta memoria es "preciosa", no solo por la posibilidad de rodar antiguos programas o juegos MsDOS en ventana que requieren mucha memoria, sino también porque el propio Windows necesita "parte" de la memoria real para cargar ciertas secciones de DLLs (normalmente DLLs de 16 bytes heredadas del 3.1), pero de todas maneras, influye negativamente en las prestaciones de Windows el tener "poca" memoria real.

3.2.4. **DEVICE= ; DEVICEHIGH=**

Con estas líneas cargamos un driver de dispositivos. Ya hemos comentado que Windows, excepto el HIMEM.SYS y el EMM386, NO necesita ninguno. Y no es conveniente tener *ninguno*.

El poner alguno, indica que es un "viejo" programa o necesario para un "viejo" dispositivo. Por tanto esto influye negativamente en el comportamiento de Windows.

En caso de tener que poner algún driver (solo en caso de "extrema necesidad), podemos utilizar el DEVICEHIGH (en vez del DEVICE), en este caso y una vez que además hemos utilizado el HIMEM y el EMM386.

Casi (pero no todos) todos los drivers admiten funcionar en memoria alta. Existen específicos (sobre todo los de acceso a las antiguas tarjetas SCSI, y alguno de red en modo real), que no funcionarían o provocarían cuelgues aleatorios si los cargamos en memoria alta.

La instrucción DEVICEHIGH admite además la "región" de carga (con el parámetro /L, pero es opcional). Se llama "región" de carga, al número del posible hueco en memoria superior. Recordar que comentamos que hay huecos en la memoria superior que se pueden utilizar, si hemos puesto el EMM386 con el parámetro RAM y utilizamos DOS=UMB. Evidentemente,

como en teoría podemos tener varias "bios" de varias tarjetas en esos huecos de memoria, puede que no queden "contiguos". Si no quedan contiguos, admiten "numeración". Pues bien, podríamos especificar así el "numero" del hueco UMB.

3.2.5. **DOS=HIGH|LOW[,UMB|,NOUMB][,AUTO|,NOAUTO]**

No conviene andar jugando con este parámetro. Únicamente poner DOS=HIGH,UMB si utilizamos EMM386.

El intentar forzar la memoria de otra manera es únicamente para alguna situación muy específica (y totalmente improbable).

3.2.6. **DRVPARM=**

Únicamente a utilizar para disqueteras *no* estándar. En este caso, normalmente el fabricante nos indicará que parámetros debemos especificar aquí. Evidentemente se cae de su peso, que no debemos tocar esto para las disqueteras estándar que nos dan en nuestro PC.

3.2.7. **FCBS=x ; FCBSHIGH=x**

Es un parámetro viejísimo. Herencia del MsDOS 1.0 (del año 82). Este MsDOS abría los archivos mediante la técnica de "File Control Block". Es decir era necesario crear en memoria una estructura de control y pasárselo a las funciones de acceso a disco. Esto ya no se utiliza (desde hace más de 15 años). Pero por cuestiones de compatibilidad con los posibles programas MsDOS del año 82 que nos quedasen, todavía existe esta opción.

3.2.8. **FILES=nn ; FILESHIGH=nn**

Indica el número máximo de archivos que puede tener abierta una aplicación MsDOS.

Mismos comentarios que he realizado para los BUFFERS:

No tiene ningún efecto bajo Windows ni bajo las ventanas MsDOS bajo Windows. Únicamente tiene efecto en MsDOS puro. Y también tiene efecto al "arrancar" Windows, antes de que este entre en modo protegido. Antes de que entre el procesador en ese modo, se está utilizando todavía el antiguo sistema de apertura y carga de ficheros en MsDOS. Por tanto se ahorraría "algo" de tiempo al arrancar Windows, si este parámetro fuese elevado.

Mi consejo es NO ponerlo. Solo sirve para ahorrar unos segundos en el arranque de Windows. Por contra, nos disminuirá muchísimo nuestra memoria real. Recordar que esta memoria es "preciosa", no solo por la posibilidad de rodar antiguos programas o juegos MsDOS en ventana que requieren mucha memoria, sino también porque el propio Windows necesita "parte" de la memoria real para cargar ciertas secciones de DLLs (normalmente DLLs de 16 bytes heredadas del 3.1), pero de todas maneras, influye negativamente en las prestaciones de Windows el tener "poca" memoria real.

3.2.9. INSTALL= ; INSTALHIGH=

Equivale exactamente lo mismo que cargar un programa (no un driver de dispositivo), en el AUTOEXEC. Si estas líneas se ponen, se ejecutarán siempre al final del config, independientemente de en donde las hayamos situado.

3.2.10.LASTDRIVE= ; LASTDRIVEHIGH=

Indica cual es nuestra "ultima" unidad o letra de disco. Este parámetro afecta a Windows, por lo que no conviene tocarlo. En el antiguo MsDOS 6.22, si no lo poníamos por defecto, era la F:. En el MsDOS de Windows 95 / 98, por defecto es la letra Z:, por tanto no debe tocarse.

3.2.11.NUMLOCK=[ON|OFF]

Indica si queremos que la tecla de bloqueo del teclado numérico esté o no activa. No es necesario ponerlo y se asumirá lo que esté definido en la bios de nuestra máquina.

3.2.12.REM

Escrito por delante de cualquier línea, la convierte en línea de "comentarios". Por tanto no se ejecutará la instrucción que va a continuación.

3.2.13.SET variable=xxxxxxx

Permite especificar variables de entorno. En los antiguos MsDOS, esto era posible únicamente en el AUTOEXEC.BAT. Actualmente es posible en ambos sitios: en el config y en el autoexec.

3.2.14.SHELL=[[disco:]path]programa [parametros]

Este comando *si* es importante. No es necesario, pero algunas veces nos puede ayudar a solucionar algún problema.

Aquí se define el "interprete de comandos". Normalmente sabemos que dicho interprete es el COMMAND.COM pero puede ser perfectamente otro interprete que no sea de MS (existen interpretes de comandos de terceros).

Por defecto, si no ponemos la línea, asume que es el COMMAND.COM. Pero recordar que el command.com, admite parámetros. Podéis verlo dando command /? en una ventana MsDOS. Imaginar que queréis que por defecto TODOS los command (o ventanas MsDOS) de vuestra maquina se abran con unas determinadas características de tamaño de entorno ,etc... Bien en este caso, como queremos "todas", lo mas cómodo es especificarlos en el SHELL del config.sys.

3.2.15.STACKS= ; STACKSHIGH=

Define el número de stacks (pilas) para el uso internos del MsDOS. No es conveniente ponerlo y no tiene ningún efecto en Windows ni en ventanas MsDOS bajo Windows.

3.2.16. SWITCHES= /F /K /N /E[:n]

/K Fuerza un teclado "enhanced" como teclado normal.

/N Deja inactivas durante el arranque las teclas F5 y F8 que nos permite "pasar" del archivo de comandos.

/E[:n] Si se usa sin el parámetro :n indica que se debe suprimir la reasignación de ciertas extensiones de la bios (EBIOS). No es conveniente tocar este parámetro ya que puede tener efectos negativos en la memoria real de la maquina al forzar la carga de las extensiones de la bios en memoria baja.

4. AutoExec.Bat

4.1. Parámetros de configuración

Como introducción (y casi, casi como único resumen), podemos decir que el autoexec.bat se encarga, o puede encargarse de ejecutar cualquier programa MsDOS, así como de establecer las condiciones de "entorno" (ya las veremos mas adelante), de todo el MsDOS, y lo que es más importante: de todo Windows. Esto ultimo, deriva de que Windows "hereda" todo el entorno que tenía al arrancar.

Estrictamente este archivo no es necesario (al igual que el config.sys), pero por desgracia, el MsDOS y el Windows, están pensados para configuraciones regionales USA (así como el teclado). Por tanto como nuestro sistema (y nuestro teclado) no está en USA, debemos incorporar unas pocas líneas, tanto en el config, como en el autoexec.

Recordemos que en el config eran:

```
device=C:\WINDOWS\COMMAND\display.sys con=(ega,,1)
```

```
Country=034,850,C:\WINDOWS\COMMAND\country.sys
```

Y en el autoexec.bat, son:

```
mode con codepage prepare=((850) C:\WINDOWS\COMMAND\ega.cpi)
```

```
mode con codepage select=850
```

```
keyb sp,,C:\WINDOWS\COMMAND\keyboard.sys
```

(En ambas me estoy refiriendo a "España" y teclado Español. Para configuraciones en Latino América, estas líneas variarán ligeramente)

Básicamente, en estas líneas, estoy configurando la tabla de códigos, como la 850, el país España como 034, y el teclado como español "sp".

NOTA: Es importante, sobre todo si vamos a tener más de un sistema operativo, es decir, si nuestro sistema va a convivir con NT 4 o con Windows 2000, que estas líneas estén correctamente definidas.

El problema nos puede venir causado, porque si no tenemos correcta la página de códigos en Windows 98, los caracteres acentuados y caracteres locales (como la "ñ") aunque la veamos correctamente en pantalla, se almacenan con la tabla de códigos por defecto del MsDOS.

Entonces veríamos correctamente los nombres de archivos acentuados, pero internamente el nombre estaría almacenado con otro código.

En esta situación, si instalamos Windows NT o Windows 2000, estos, al configurarse (recordar que son independientes y no se apoyan en el MsDOS), se configuran con la tabla correcta de códigos: 850. Por tanto los caracteres acentuados en nombres de archivos, serán otros. De esta manera, un scandisk desde Windows 98 a la partición NT, nos dará errores en los nombres de archivo, y lo que es mas grave: intentará arreglarlos (estropeando el correcto desde "su" sistema). Exactamente igual nos pasará desde Windows 2000.

4.2. *Condiciones de entorno*

Se entiende por entorno, aquellas variables que son comunes a todo el sistema. Se heredan entre los procesos.

En general no son necesarias, excepto para programas particulares que las vayan a utilizar.

Las variables que queramos que se vean en "todo" el sistema y que puedan ser leídas por un programa, se asignan en el autoexec mediante el comando SET. Esto lo utilizan muchas aplicaciones.

La variable más curiosa de entorno, es la TEMP. Esta la utilizan desde el comienzo del MsDOS, muchas aplicaciones, y en particular, también la utiliza Windows.

Por defecto, sino está definida, el propio MsDOS le asigna el contenido C:\WINDOWS\TEMP y por costumbre desde los inicios del DOS, esta carpeta se utiliza para escribir en ella ficheros temporales que necesiten las aplicaciones, y que por definición pueden ser borrados en cualquier momento.

Los programas o aplicaciones bien realizadas, deberían además ser las responsables de borrarlos. Pero esto, quizá sea mucho pedir...

Tal y como estabamos comentando con el entorno, nosotros podemos definir en el autoexec otra localización de la carpeta TEMP. Lo más normal es tener:

```
SET TEMP=C:\TEMP
```

Y a su vez tener creada la carpeta TEMP en C:.

Una de las variables de entorno más importante en el PATH (camino). Cuando tecleamos un programa para su ejecución, tanto el MsDOS como Windows, buscan el programa en la carpeta en donde estamos en ese momento, y si no lo encuentran, lo buscan en el "camino" que esté definido en nuestra variable PATH.

Por defecto, sino especificamos un path, en Windows por defecto el path es C:\WINDOWS;C:\WINDOWS\COMMAND (se debe tomar nota, que los distintos caminos, se separan por punto y coma).

Como PATH, además, es una variable de entorno, puede ser asignada mediante el comando SET. Es lo más cómodo. Imaginar que queremos "añadir" al PATH que tuviésemos en un momento determinado, la carpeta C:\KK. Bien, lo más sencillo sería escribir:

```
SET PATH=%PATH%;C:\KK
```

Fijaros que lo que estamos haciendo, es decirle que el nuevo PATH, es igual al anterior (en este caso, se pone %PATH% -es decir encerrado entre símbolos %), y a continuación, separado por punto y coma, el camino que queremos añadir.

NOTA 1: Para ver el PATH que tenemos en un determinado momento, podemos abrir una ventana MsDOS y teclear simplemente PATH. Esto nos mostrará el contenido del PATH. Igualmente en una ventana MsDOS, si tecleamos el comando

SET nos mostrará todas las variables de entorno, y en particular el propio PATH, ya que esta es una variable de entorno.

NOTA 2: Cuando dentro de Windows, vamos a Inicio->Ejecutar y tecleamos el nombre de un programa, Windows primero buscará en C:\WINDOWS\SYSTEM y si no lo encuentra, a continuación buscará en el PATH.

4.3. Empieza la optimización de nuestro sistema.

Recordar que el directorio de archivos temporales, "debe" estar vacío. Si tiene muchos archivos, degrada de una manera apreciable las prestaciones y velocidad de Windows. Es conveniente borrarlo de vez en cuando. Y ahora la pregunta del millón: ¿no podríamos utilizar el propio autoexec.bat, para que cada vez que arranquemos, sea él el encargado de "limpiar" esta carpeta?.

Pues sí podemos y "debemos" hacerlo.

Una manera muy sencilla de hacerlo, es incorporar las siguiente líneas de código en nuestro autoexec.

```
if not exist c:\temp\*. * goto cont0  
attrib c:\temp\*. * -s -h -r  
echo S | del c:\temp\*. * >nul  
:cont0
```

Esto nos borrará el contenido de la carpeta C:\TEMP. Si tuviésemos los temporales en otra carpeta (por ejemplo, en la carpeta por defecto de Windows C:\WINDOWS\TEMP), deberemos sustituir C:\TEMP por el nombre de la carpeta que tuviésemos los temporales en nuestro sistema.

Es importante recordar además que debe sustituirse la "S" de la línea "echo S", por una "Y" (sin las comillas) si tuviésemos Windows en Inglés.

4.4. Confusión existente en torno al MSCDEX

En muchas consultas realizadas y debido a cierta confusión que existe con el MSCDEX y la posibilidad de ver o no nuestra CDROM desde MsDOS puro, voy a hablar un poco sobre este tema.

Notas a tener en cuenta:

- 1) No le hace falta a Windows, que tengamos definido "nada" en nuestro config y autoexec, para ser capaz de ver la CDROM.
- 2) Si tuviésemos algo definido, es mas que probable que a Windows no le quede mas remedio que acceder a nuestra CDROM, utilizando el viejo método de acceso de 16 bits, perdiendo entonces la capacidad de acceso en 32 bits. Además, también es mas que probable, que si nuestra CDROM fuese IDE, perdamos el acceso a 32 bits también en nuestro disco duro si este está en el mismo cana IDE.

Con lo anterior, quiere decir: "cuidadito" con lo que tenemos o ponemos allí.

Repasemos un poco el antiguo MsDOS, para entender como se accedía a una CDROM (sistema de 16 bits).

Para acceder a la CDROM, necesitamos dos componentes software:

- 1) Un driver de dispositivo (por ser driver, debe estar en el config.sys), que nos permite ver un dispositivo de tipo "stream" (o de flujo) como un dispositivo "récord" (orientado a registro). Recordad que un CDROM es un dispositivo "stream" - como si fuese una unidad de cinta -.
- 2) Un programa (por tanto, montado en el autoexec), que sea capaz de acceder al dispositivo virtual montado en el punto 1), y devolvernos los datos como si fuese una unidad de "disco". Este programa es un estándar de Microsoft: el MSCDEX (pero podría ser cualquier otro y de echo existieron algunos durante la vida del antiguo MsDOS).

A partir de ahora me voy a referir únicamente a los CDROM IDE. (la idea básica, de todas maneras, es igualmente extrapolable a los CDROM SCSI).

Hasta que surgió Windows 98, y nos incorporó un driver casi "universal" para todas las unidades de CDROM, era bastante normal en el MsDOS el tener drivers del tipo:

```
DEVICE=C:\HITACHI.SYS /D:MSCD001 .... o  
DEVICE=C:\PIOONER.SYS /D:.....
```

Es decir un driver de nuestro fabricante de CDROM (normalmente lo identificamos por el parámetro /D:MSCD001) en el config.sys para poder ver nuestra CDROM. Igualmente teníamos una línea del tipo MSCDEX /D:MSCD001 en el autoexec.

Fijaros que el nombre puesto en /D:MSCD001, puede ser cualquiera, con tal de que sea el mismo en el driver del config y en el programa MSCDEX.

Bien, retomando el tema, estabamos diciendo que Windows nos aporta un driver "casi" universal: el OAKCDROM.SYS (que podemos encontrarlo en el disco de inicio de Windows 98, o bien en la carpeta C:\WINDOWS\COMMAND\EBD). Por tanto incorporando este driver en el config y a su vez invocando al programa MSCDEX, tendríamos acceso a la CDROM en modo MsDOS puro.

Pero... debido a los problemas que he comentado al principio de este capítulo, esto está completamente desaconsejado. Perderíamos muchas prestaciones en nuestro sistema.

Entonces, la pregunta es ¿cómo puedo acceder a la CDROM, al reiniciar en modo MsDOS, desde Windows?. Bueno, y además, existe un problema: un driver de dispositivo, "debe" cargarse en el config.sys. Veamos tres posibilidades para solucionar este problema:

- 1) Cargar el OAKCDROM.SYS en el config de nuestro propio Windows. Y *NO* cargar el MSCDEX. Con esto evitamos que el acceso se haga a 16 bits.
Posteriormente creamos un archivo en nuestro directorio de Windows, llamado DOSSTART.BAT que únicamente tuviese la línea de MSCDEX /D:MSCD001. Si este el archivo ya existiese, incorporarle dicha línea.

Esta solución funciona, pero no me gusta por dos motivos: uno, consumo de memoria MsDOS (al cargar el driver anterior), que luego no sirve para nada bajo Windows. Y segundo motivo, este tipo de drivers en combinación con ciertas controladoras y unidades de CDROM, pueden causar inestabilidades al Windows.

- 2) Utilizar algún programa de los llamados "cargadores" de drivers. Por ejemplo, Creative Labs, tenía en su servidor FTP, un programa llamado CTLOAD, que era capaz de cargar un driver una vez que estuviésemos en MsDOS y sin necesidad de incorporarlo en el config. La manera de cargarlo sería entonces:

```
CTLOAD C:\WINDOWS\COMMAND\EBD\OAKCDROM.SYS /D:MSCD001
MSCDEX /D:MSCD001
```

(y suponiendo que el programa CTLOAD, lo hemos dejado por ejemplo en C:\windows\command para que lo encuentre en el path).

Incorporando estas líneas en el DOSSTART.BAT citado anteriormente, tendríamos acceso a la CDROM al reiniciar en modo MsDOS.

- 3) Tercera posibilidad: utilizar un config y autoexec propio y crear un acceso directo a un "command.com" desde el escritorio con ese config y autoexec propio que incorpore esas líneas cada una en su correspondiente archivo. Tampoco me gusta, porque esto implica el tener que "mantener" otros config y autoexec.

La solución mas "limpia" en mi opinión es la 2).

5. ¿Cómo utiliza el Registro Windows98?

El registro contiene pares de datos, llamados claves y valores y se manipulan con lo que en programación se conoce como API's (Application Programs Interfaces) de registro Win32.

Hay claves dinámicas que apuntan a lugares concretos de la memoria o a cualquier función de retrollamada. Estas las utilizan los controladores de dispositivos (Drivers) y los subsistemas de Windows (Subsistema gráfico por e.j.) para guardar datos de tipo dinámico, es decir, algo que cambia.

Para la creación de claves se pueden utilizar archivos INI, mediante Scripts o con el editor del registro "Regedit".

Windows 98 verifica y almacena información de configuración en el registro en gran parte de los parámetros que contiene durante el inicio del sistema.

5.1. *Las herramientas*

5.1.1. **Un comprobador: Scanreg y Scanregw.**

El primero para utilizarlo en modo MS-DOS (real) y el segundo dentro de la interfaz de Windows (Modo protegido).

Entre sus funciones de mantenimiento del sistema es localizar y corregir errores del registro. Al iniciarse el ordenador examina el registro en busca de estructuras incompletas, si no las hay realiza una copia de seguridad para cada día (no en cada inicio). En caso de detectar errores puede recuperar una de estas copias de seguridad. Aunque sólo mantiene cinco copias puede configurarse para que almacene más. Si no encontrara copias de seguri-

dad intentaría corregir el registro. Y también reduce su tamaño eliminando el espacio no utilizado.

5.1.2. Un editor: Regedit.

Su función es ver y editar el registro. por supuesto podemos también modificar.

Un consejo!! hacer una copia del registro antes de cambiarle valores.

El regedit, en caso necesario también puede ejecutarse en MS-DOS.

Tenemos dos editores más que influyen en el registro y que no vienen instalados por defecto, han de ser habilitados y/o instalados a propósito.

El editor de perfiles y el editor de planes de sistema.

6. Estructura y tipos de datos del registro

Componen el registro las claves, los valores y los datos.

6.1. Las claves

Está estructurada jerárquicamente. Primero las claves y cada clave contiene una o más subclaves o uno o más valores.

Cuando lo editamos con el Regedit se nos muestran seis subárboles al más puro estilo del explorador de Windows.

Las claves raíces siempre comienzan por la cláusula "HKEY_" para indicar que es un identificativo único, denominado manejador.

Las raíces contienen a su vez más subclaves y estas más.... son anidaciones.

Las claves y subclaves pueden contener caracteres visibles, la única excepción es la barra invertida "\", y no hacen distinción entre mayúsculas y minúsculas aunque si las reconocen.

6.2. Los valores

Los valores contienen datos, y pueden ser de tres maneras; Texto, Binario o DWORD.

- **Texto** - Caracteres de longitud variable finalizando en nulo.
- **Binario** - Hexadecimales de longitud variable.
- **DWORD** - Valor en hexadecimal de 32 bits y 8 dígitos.

6.3. Las raíces

Las raíces son: HKEY_LOCAL_MACHINE y HKEY_USERS

Si ya sé que se nos muestran 6, pero las otras cuatro son extracciones de éstas dos.

- 1) **HKEY_LOCAL_MACHINE**
 - 1.1) **HKEY_CURRENT_CONFIG**
 - 1.2) **HKEY_CLASSES_ROOT**
 - 1.3) **HKEY_DYN_DATA**
- 2) **HKEY_USERS**
 - 2.1) **HKEY_CURRENT_USER**

Vistas de modo genérico (en cada equipo pueden variar mucho en cuestión de subclaves y valores) diremos que:

HKEY_LOCAL_MACHINE en esta se almacena la información específica de usuario pero referente al tipo de hardware instalado y parámetros del software. Esto es usado por TODOS los usuarios que inician sesión.

En sus ramificaciones tenemos:

HKEY_CURRENT_CONFIG, encargada de gestionar el Plug&Play, tiene la configuración ACTUAL de un equipo con configuración de hardware múltiple.

HKEY_CLASSES_ROOT, esta clave apunta a la principal

HKEY_DYN_DATA, son datos dinámicos (para la RAM del sistema). La información cambia conforme se agregan o eliminan dispositivos. El "Administrador de dispositivos" utiliza estos datos para mostrarnos la configuración actual y actualizan de forma continua al "Monitor de Sistema".

HKEY_LOCAL_MACHINE\Software\Classes, describe ciertas configuraciones del software. OLE y asociaciones para arrastrar y soltar (drag&drop) accesos directos y aspectos de la interfaz de windows98.

HKEY_USERS. Toda la información de todos los usuarios que inician sesión, tanto genérica como específica. Se compone de parámetros predeterminados de aplicaciones, configuraciones del escritorio, etc...

Contiene a su vez subclaves para cada usuario.

Y finalmente

HKEY_CURRENT_USER, que apunta a una de las ramas de HKEY_USERS, claro, esto es para el usuario que está conectado en ese momento.

7. Las claves del registro

El registro contiene claves con valores que cambian para cada usuario y sistema, y es imposible describir todas las posibilidades, pero aun así veremos las subclaves más significativas.

Las Handle Keys o HKEY_ es lo que marca a una clave como raíz. (¿Clave de manejo?)

7.1. *HKEY_CLASSES_ROOT*

Sabemos que la HKEY_CLASSES_ROOT se almacena en System.dat y por tanto están los datos para compatibilidad con DDE y OLE de Windows 3.x. Además contiene los nombres de los archivos registrados, sus iconos, órdenes, etc. E información de los viewers (visualizadores) gráficos, los manejadores de las hojas de propiedades y los ActiveX. Normalmente la parte de configuración de cualquier programa registra la extensión de sus archivos y las órdenes que se le aplican.

Esta clave apunta a la principal HKEY_LOCAL_MACHINE \Software\Classes.

Se tiene dos tipos, Claves de extensión de los nombres de archivo y las claves de definición de clases. Una aplicación que tenga soporte DDE tendrá en la subclave "Shell" otras subclaves como open o print. Aquí contendrán la ruta de la aplicación y orden para ejecutarse. Un txt contendrá c:\windows\notepad.exe "%1".

Todas las clases se identifican por un punto y tres caracteres ".bmp .jpg .txt .doc ."
Cuando hacemos clic sobre un archivo, Windows 98 buscará una clave que coincida con la extensión del mismo.

Los CLSID son las propiedades de objetos Activex.

7.2. **HKEY_CURRENT_USER**

HKEY_CURRENT_USER, que como sabemos es el usuario conectado.

Aquí encontraremos las subclaves:

- **AppEvents** - Rutas de acceso y archivos de sonidos para los sucesos del sistema, en EventLabels se encuentran las etiquetas para estos y Schemes el archivo .wav concreto.
- **Control Panel** - Todas las subclaves del panel de control.
- **InstallLocationMRU** - Localización desde donde fueron instaladas las aplicaciones más recientes. (MRU = Most Recently Used) Keyboard Layout - Subclaves de la configuración del teclado.
- **Network** - Conexiones de red persistentes y recientes.
- **RemoteAccess** - Direcciones y perfiles de acceso remoto a red.
- **Software** - Toda la configuración software del usuario, con información de todas las aplicaciones.

7.3. **HKEY_LOCAL_MACHINE**

• **Config**

La configuración actual de la máquina y que viene en la subclave \Config. Contendrá múltiples configuraciones con su ID (identificador único) y que el ordenador usará en su conexión a red o desacoplado. Cada configuración es una subclave que cuelga de \Config.

Además encontraremos:

- **Enum** - Como una Enumeración del hardware instalado en el sistema. Todos los dispositivos contendrán el tipo, la letra de unidad asignada, el ID de hardware y el fabricante y la información relacionada con el controlador. Algunas de sus subclaves, ESDI (discos fijos), FLOP (flexibles), ISAPNP (P&P ISA), etc.
- **Hardware** - Puertos serie y modems utilizados por Hyperterminal.
- **Network** - Información de red.
- **Security** - Acceso de seguridad.
- **Software** - Información específica del ordenador acerca del software instalado. Y que puede escribir en el registro.
- **System** - Control del inicio del sistema, la carga de controladores de dispositivos, servicios de Windows y el comportamiento de windows98. Son conjuntos de control y contiene los parámetros para controladores y servicios que se puedan cargar en Windows, están bajo CurrentControlSet, y aquí en Control y Services, La primera es para el inicio del sistema y la segunda dispone de la información de los controladores de dispositivos kernel (ya sa-

bréis que es el modo "0") y subclaves con descripciones estáticas del hardware al que se le acoplarán los controladores.

Debajo de control:

- **Computername** - el nombre del ordenador
- **FileSystem** - Sistema de archivos.
- **IdconfigDB** - Su identificación de configuración.
- **Keyboard layouts** - Lista de librerías dinámicas (dll) referentes al idioma.
- **Resources** - Los controladores Multimedia
- **NetworkProvider** - Los proveedores de red
- **Nls** - Idioma, localización.
- **PerfStats** - Estadísticas recogidas para ver por el monitor de sistema
- **Print** - Las impresoras.
- **SessionManager** - Variables globales, más una lista de aplicaciones que no se ejecutan bien en Windows 98, dll que deberían verificarse y directorios y nombres de archivo para todas las dll.
- **TimeZoneInformation** - Configuración de la zona horaria.
- **Update** - Sólo dice si la instalación ha sido actualización o limpia.
- **VMM32** - Los nombres de los archivos Vxd combinados con el dispositivo virtual Vmm32.vxd.

Y en services:

- **Nombre_agente** - subclaves para cada agente instalado, monitor de red, copia seguridad de red, etc.
- **Arbitrators** - Subclaves para los árbitros que se requieren para gestionar los recursos entre dispositivos. (como los de direcciones, DMA, E/S e IRQ)
- **Class** - Clase de dispositivos que permite el S.O.; unidades de disco, teclado, pantalla, ratón,.
- **MSNP32-NWNP32** - Subclaves de proveedores de red en modo protegido, su información de seguridad y de inicio de sesión del proveedor.
- **VxD** - Controladores virtuales de dispositivos, tales como unidades de disco, red, caché de disco.

7.4. **KKEY_USERS**

KKEY_USERS contiene todos los datos predeterminados de usuario en la subclave .DEFAULT, además todos los de los usuarios que se han conectado. La información de .DEFAULT es la base para crear un perfil de usuario, a uno que no tenía todavía perfil.

Dentro de .DEFAULT encontraremos entre otros:

- AppEvents
- Control Panel
- RunMRU
- Software

Si sólo existe .DEFAULT, la HKEY_CURRENT_USER apunta a ella, si existe otra con nombre de usuario entonces apuntará a la segunda.

En caso de similitud de datos en HKEY_LOCAL_MACHINE y HKEY_CURRENT_USER, la segunda tiene preferencia sobre la primera. Primarán datos y configuraciones de usuario sobre los valores predeterminados.

7.5. HKEY_CURRENT_CONFIG

Almacenada en system.dat, apunta a la configuración actual del sistema, que se encuentra en HKEY_LOCAL_MACHINE\Config\000x, en donde x es el número de configuración.(0001, 0002 ...)

- **Display** es la subclave de pantalla
- **Enum** las configuraciones de BIOS P&P
- **System** tiene las impresoras disponibles.

7.6. HKEY_DYN_DATA

La información de configuración que debe almacenarse en RAM, susceptible de modificación rápidamente. La información se crea cada vez que se inicia Windows 98. Así si la editamos con Regedit siempre estará actualizada. Nos mostrará todos los dispositivos, incluso los que no se hayan cargado por problemas.

8. El Registro de Windows 98 comparado con Windows 95 y NT/2000

No existen diferencias apreciables entre el Registro de Windows 98 y el de Windows 95, tienen la misma organización y el mismo Editor del Registro.

Pero el primero ha sido optimizado en la estructura de su código y datos que lo implementan, así se ha conseguido que sea más rápido.

En cuanto al NT/2000 son muy parecidos, pero las diferencias se encuentran en HKEY_CLASSES_ROOT y en HKEY_USERS,

- NT/2000 almacena el Registro en colmenas, Windows 98 lo hace en archivos binarios.
- NT/2000 implementa seguridad total en las colmenas y sus claves individuales, Windows 98 carece de seguridad o ésta es mínima.
- HKEY_LOCAL_MACHINE\System es extremadamente distinta de uno al otro.
- NT/2000 tiene otro Editor del Registro, el cual aprovecha la seguridad y los tipos diferentes de datos susceptibles de ser almacenados en un valor.

9. Métodos de introducir claves en el registro (archivos REG e INF)

9.1. Archivo .Reg

Creamos un fichero .REG con una sintaxis predeterminada.

Una clave que cuelga de, HKEY_LOCAL_MACHINE Software y que a su vez tiene los distintos posibles valores del registro, así como dos subclaves. Una con "algo" de contenido y la otra totalmente vacía.

Deberemos crear un fichero .REG con lo siguiente:

```
REGEDIT4
[HKEY_LOCAL_MACHINE\Software\nombreclave]
@="Con texto por defecto"
"CadenaCaracteres"="pongo lo que quiero"
"ValorBinario"=hex:00,01,02
"DoblePalabra"=dword:00001234
[HKEY_LOCAL_MACHINE\Software\nombreclave\nombreSubclave]
@="Solo con el Defecto"
[HKEY_LOCAL_MACHINE\Software\nombreclave\SubclaveVacía]
```

Se ejecuta con el botón derecho y combinar.

Deberían salir dos mensajes, el primero solicitando autorización para introducir los datos en el registro y el segundo advirtiéndole que ha sido introducido o el error consecuente.

9.2. Archivo .Inf

Creando un .INF. Cambia la sintaxis. Además la ejecución, se hace igualmente con el botón derecho, pero ahora hay que darle a "instalar".

```
[versión]
signature="$CHICAGO$"
[DefaultInstall]
AddReg=Enable.nombreclave
[Enable.NombreClave]
HKLM,Software\nombreclave,,0,"Con Texto por defecto"
HKLM,Software\nombreclave,"CadenaCaracteres",0,"pongo lo que quiero"
HKLM,Software\nombreclave,"ValorBinario",1,00,01,02
HKLM,Software\nombreclave\nombreSubclave,,0,"Solo con el Defecto"
HKLM,Software\nombreclave\SubclaveVacía,,0,""
```

En ambos casos si la clave ya existe lo que haríamos es modificar sus valores.

10. El Registro y el P&P

Plug and play, conectar y listo en español, son un conjunto de especificaciones, con las que se pretende que el ordenador, los dispositivos, controladores y sistema operativo trabajen conjuntamente sin la necesidad de la intervención del usuario.

Para ello se necesita que todos los componentes cumpla esa especificación P&P.

- Un sistema operativo P&P
- BIOS P&P o en su caso BIOS ACPI. (Basic In Out System; Advanced Configuration Programed Interface)

- Dispositivos P&P y sus controladores (drivers)

La eficacia se verá resentida dependiendo de estos tres componentes. El caso de que ninguno sea P&P, el sistema carece de dinamismo, y las tarjetas deben configurarse a mano y sus controladores cargarse manualmente también.

En el caso de Windows98, tenemos un S.O. que sí es P&P, y que puede operar dinámicamente. La configuración de los dispositivos puede resultar muy sencilla, y los controladores cargarse automáticamente.

Y en un caso que los tres componentes sean P&P, la configuración es tan simple como conectar el dispositivo e iniciar el ordenador.

El registro contiene la base de las especificaciones P&P, con ello el usuario conecta un nuevo dispositivo y Windows 98 debe configurarlo automáticamente, en todo caso pedirá los drivers si no los incluye el propio S.O.

Cuando conectamos un dispositivo nuevo, Windows 98 utilizará el ID del mismo para buscar los archivos.inf que le pertenezcan. Crea una entrada en HKEY_LOCAL_MACHINE y traslada la información del .inf a la clave creada.

- En la clave **Enum de HKEY_LOCAL_MACHINE** se hará referencia a la enumeración que Windows 98 ha comprobado del dispositivo.
- En **Enum de HKEY_CURRENT_CONFIG** se contendrá varias subclaves que especifican la BIOS P&P y otros elementos.
- La **Enum de HKEY_DYN_DATA** contendrá el bus ISA P&P.
- Y en **HKEY_DYN_DATA\Config\Manager\Enum** se definirá la asignación del propio recurso, sus problemas y su configuración.
- **HKEY_LOCAL_MACHINE\Enum** contendrá pues las necesarias subclaves del hardware utilizado por el ordenador. Windows 98 usará esta información para la asignación de recursos, E/S, IRQ, etc...

Todos los valores son específicos de los fabricantes y sus dispositivos, por ello varían en una u otra máquina.

Aún así, le echamos un vistazo:

- La **subclave BIOS** contiene las entradas de los componentes P&P de la propia BIOS, temporizadores, controladores, DMA. Cada subclave comienza por la cadena *PNP seguido de un número de cuatro dígitos que representa clases por las que se agrupan los componentes, *PNP0000. Estas contienen a su vez subclaves (00, 01) con datos sobre el nombre de clase, descripción del dispositivo, su nombre, y un ID (único) de hardware y que servirán para su configuración.

Valores:

PNP0000-PNP0004 Controladores de interrupciones.
 PNP0100-PNP0102 Temporizadores.
 PNP0200-PNP0202 Controladores DMA (Direct Memory Acces)
 PNP0300-PNP0313 Controladores del teclado.
 PNP0400-PNP0401 Puertos LPT
 PNP0500-PNP0501 Puertos Serie
 PNP0600-PNP0602 Controladores HDD
 PNP0700 Controladores disquete estándar

PNP0800 Altavoz
PNP0900-PNP0915 Tarjetas gráficas
PNP0A00-PNP0A04 Buses
PNP0B00 Reloj de la CMOS en tiempo real.
PNP0C01 Extensiones P&P para BIOS
PNP0C04 Procesador de datos numéricos
PNP0E00-PNP0E02 PCMCIA
PNP0F01 Mouse serie
PNP0F00-PNP0F13 Puertos Mouse
PNP8xxx Tarjetas de RED
PNPA030 Controladores CD-ROM
PNPB0xx Otros Adaptadores

- **La subclave ESDI** contendrá configuración del controlador IDE
- **La FLOP**, los de los floppys.
- **ISAPNP**, las enumeraciones del bus ISA P&P.
- **MF**, otra información específica del fabricante.
- **Monitor**, información sobre el monitor, si el sistema no detecta el tipo se exige al usuario una elección.
- **Network**, RED.
- **PCI**, entradas de las tarjetas pinchadas, en la placa, tipo PCI.
- **Root**, aquí también hay entradas tipo *PNPxxxx para dispositivos no P&P.
- **SCSI**, Dispositivos SCSI.

Los códigos se agrupan en clases de acuerdo con los cuatro dígitos, la configuración se modifica con el Administrador de dispositivos (o a mano ¿no?)

Windows 98 detecta y enumera al dispositivo y procede a su configuración.

Durante el proceso de detección se crea un archivo Detlog.txt en C:\, aquí se observan los problemas de detección de hardware.

11. Otros valores importantes del registro.

11.1. La lista MRU

Windows 98 guarda las preferencias de escritorio y ventanas que se abren en el registro, incluyendo el tamaño de iconos su posición y orden.

Hay 28 entradas se utilizarán para estas preferencias en una lista MRU.

Según se cierra una ventana se escribe una subclave, 0, 1, 2... en

HKEY_CURRENT_USER\Software\Microsoft\Windows\Currentversion\Explorer\Streams

Y el orden MRU para las 28 entradas en:

HKEY_CURRENT_USER\Software\Microsoft\Windows\Currentversion\Explorer\Streams MRU

Si el escritorio o una ventana no se incluye más en MRU, uno u otra usarán la configuración predeterminada cuando se pinten o refresquen.

Si les realizamos un cambio, las preferencias se mueven en la lista MRU, permaneciendo hasta llegar a la posición 28, si se cierra otra ventana el elemento 28 se borrará.

11.2. *Los modems*

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\Modem

Cada modem empezará con \0000 y sus subclaves adicionales, con los comandos AT y controladores.

Clave Init: Inicialización del modem, comenzará por la entrada 1 normalmente AT<cr>, la 2, etc...

Clave Responses: Cadenas que el modem enviará a Windows 98 en respuesta a las ordenes o durante la conexión.

Clave Settings: órdenes para configurar parámetros.

11.3. *TCP/IP de la subclave MSTCP*

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP

Todos los valores son de tipo DWORD y podéis echarle un vistazo.(Son bastantes)

En la clave

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP\ServiceProvider

Subclave Class y ProviderPath: resolución de servicios y las API del registro windows Sockets.

Las siguientes describirán el orden utilizado para resolver los nombres de host.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\NetTrans\000 n : Valores en referencia con los adaptadores de Red.

Los **nodos de TCP/IP** se encuentran bajo

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\MSTCP\nodeType

1 b-node Sólo difusión

2 p-node Sólo WINS

4 m-node Difusión, después WINS

8 h-node WINS, después difusión

Los predeterminados son:

Si no existe DHCP ni WINS = 1

Si no existe DHCP y WINS está habilitado manualmente = 8

Si existe DHCP y éste activa WINS = 4

Si existe DHCP y WINS está habilitado manualmente = 8

Si existe DHCP y WINS está deshabilitado = 1

12. Msdos.Sys - Parámetros de configuración

Msdos.sys es un archivo oculto de parámetros para windows 98.

Existen dos secciones : [Paths] y [Options]

12.1. [Paths]

Podemos encontrarnos con los siguientes valores, en [Paths]:

- **HostWinBootDrv** = "Disco en el que se encuentra instalado windows98"
Por defecto será C.
Nos aporta la información de en qué disco está el sistema.
- **UninstallDir** = "path o ruta para poder desinstalar windows"
Por defecto C.
Nos presenta la localización de W95undo.dat y W95undo.ini. Estos archivos son necesarios para desinstalar Windows. Requisitos: Windows se instaló como actualización de win 95 y además se escogió guardar archivos para desinstalar.
- **WinBootDir** = "Ruta al directorio de Windows"
Por defecto C:\WINDOWS
Parámetro que utilizará el cargador del sistema para iniciar Windows.
- **Windir** = "lugar especificado en la instalación"
Por defecto C:\WINDOWS (SI durante la instalación elegimos otro nombre o lugar será diferente)
Localización del directorio especificado durante la instalación.

12.2. [Options]

Puede contener o podemos incluirlas manualmente:

- **Autoscan** = "n" donde n = a un número
Controlará la ejecución de Scandisk si el sistema ha detectado un apagado incorrecto.
Por defecto = 1
Posibilidades:
0 No se ejecutará
1 Preguntar antes de ejecutar

- 2 No preguntará su ejecución, pero sí si los posibles errores han de repararse.
- **BootDelay** = <Segundos>
Defecto: 2
Cantidad de segundos en los que aparece la frase "Iniciando Windows" y por tanto permite pulsar F8, en Windows 95.
NOTA: Esta opción no está soportada en Windows 98.
 - **BootSafe** = <Booleano>
Defecto: 0
Si tiene valor 1, el PC arrancará en el "modo a prueba de fallos".
 - **BootGUI** = <Booleano>
Defecto: 1
0 Después de aplicar config.sys y autoexec.bat se quedará en el símbolo de sistema. Deberá teclearse "win" para iniciar windows.
1 arrancará Windows
 - **BootKeys** = <Booleano>
Defecto: 1
0 Desactiva las teclas de función "F4", "F5", "F6" y "F8", durante el arranque de Windows
1 Activa las teclas de función "F4", "F5", "F6" y "F8", durante el arranque de Windows
NOTA: Colocando BootKeys=0 sobrescribe el uso de BootDelay=n.
 - **BootMenu** = <Booleano>
Defecto: 0
0 Deberá pulsarse la tecla "F8" o mantener la "CTRL" durante el inicio para ver el menú de opciones.
1 Aparecerá siempre el menú de inicio.
 - **BootMenuDefault** = <Numero>
Defecto: 1 si el sistema se está ejecutando correctamente
Con valor 3 si el sistema falló en el arranque anterior.
Se utiliza para seleccionar la opción del menú de inicio de Windows que se ejecutará por defecto.
 - **BootMenuDelay** = <Numero>
Defecto: 30
Este valor es usado para colocar el número de segundos que el sistema va a esperar en el menú de inicio de Windows, antes de arrancar automáticamente sino seleccionamos ninguna opción.

NOTA: Esta opción no tiene sentido a no ser que la opción BootMenu=1 haya sido añadida en la sección [Options].

- **BootMulti** = <Booleano>
Defecto: 1
Un valor de 0, desactiva la opción de "multi-boot" o arranque del "antiguo" o previo sistema operativo.
Un valor de 1 activa tanto la tecla F4 como la posibilidad en el menú de seleccionar la opción de arrancar el sistema operativo anterior.
- **BootWarn** = <Booleano>
Defecto: 1
Un valor de 0 desactiva el aviso del modo a prueba de fallos al ejecutarse el menu inicio.
- **BootWin** = <Booleano>
Defecto: 1
Colocando un 1 fuerza a Windows a cargarse en el inicio.
Un valor de 0 desactiva Windows como su sistema operativo por defecto. (esto solo tiene sentido si teníamos instalado MS-DOS 5.x o 6.x previamente en nuestro PC)
NOTA: Presionando F4 invierte el defecto de arranque solo si BootMulti=1. (Por ejemplo, presionando F4 con una opción 0, fuerza a Windows 95/98 a arrancar).
- **DoubleBuffer** = <Booleano>
Defecto: 0
Colocando un 1 activa el "double-buffering" para los controladores que necesitan esto (por ejemplo los controladores SCSI).
Colocando un 2, es una opción incondicional que activa el "double-buffering" mirando cuando el controlador es necesario o no.
- **DBLSpace** = <Booleano>
Defecto: 1
Un valor de 1 carga automáticamente el DBLSPACE.BIN.
Un valor de 0 impide su carga.
NOTA: Windows 95 usa o Dblspace.bin o Drvspace.bin si alguno de ellos está presente en el directorio principal de C: Para desactivar esta opción si no tuviésemos discos comprimidos, debe forzarse un valor de 0.
Por ejemplo:
DBLSpace=0
DRVSpace=0

- **DRVSpace** = <Booleano>
Defecto: 1
Mismo sentido que la opción anterior pero con DRVSPACE.BIN.
- **LoadTop** = <Booleano>
Defecto: 1
Un valor de 0 no permite a windows cargar COMMAND.COM o DRVSPACE.BIN o DBLSPACE.BIN por encima de los 640 Kbs. Si tenemos problemas de compatibilidad con alguna vieja aplicacion DOS, es conveniente probar a forzar un cero en esta opción.
- **Logo** = <Booleano>
Defecto: 1
Si tiene el valor 1 hace que aparezca el logo de Windows al arrancar.
Con el valor 0 se desactiva el logo de Windows. Un valor de cero, libera una serie de interrupciones software que pudieran causar incompatibilidades con antiguos TSR del modo DOS o incompatibilidades con cierto manejadores de memoria de terceros.
- **Network** = <Booleano>
Defecto: 0
Con un valor 1, nos paparecerá la opción "Modo a prueba de fallos con soporte de RED" como una opción en el menu de inicio.

13. Vigilancia del sistema (SFC)

SFC es el acrónimo de **S**ystem **F**ile **C**hecker, ¿chequeador/testeador/comprobador de los archivos de sistema?

Esta herramienta es muy valiosa si se le lleva un seguimiento.

Reside en c:\windows\system como SFC.EXE.

Para ejecutarlo sólo se necesita escribir SFC en Ejecutar del botón inicio.

Cuando se ejecuta por primera vez (debería hacerse al terminar de instalar Windows98 y antes de cualquier instalación de aplicación/programa/driver), crea una base de datos, en la que guarda la versión de un gran número de archivos, incluidas todas las librerías dinámicas (dll) del PC.

Una vez ejecutado, marcamos las dos casillas de la parte inferior de la ventana de configuración.

Después, siempre que lo ejecutemos, comparará su base de datos con las actuales, y que en caso de diferir nos permitirá, o darlo como válido o recuperar la correcta.

Suele darnos grandes sorpresas, cuando comprobamos las dll después de instalar algún programa/aplicación, ver como se nos machacan versiones más modernas por otras antiguas, que al final desembocan en la inestabilidad del sistema.

Pero también tiene un pequeño BUG que hay que tener en cuenta:

Si encuentra dañado el USER.EXE o KENL.EXE, NO DEBEMOS RECUPERARLOS CON SFC, por una parte porque suele equivocarse (por cuestiones técnicas) y no están dañados y por otra porque los recuperará del CD del primer lugar en que los encuentre y el primer lugar tiene ambos archivos minimizados para la instalación y no son los correctos, si los hubiésemos recuperado el sistema no arrancará. Hay que decirle que no, y manualmente recuperar (en caso de asegurarnos que sí está dañado).

Desde una sesión MS-DOS:

```
c:  
cd \windows\system  
extract /a /e e:\win98\base5.cab user.exe
```

También debemos seguir unas pautas en la comprobación de versiones:

Nunca nos fijaremos en la fecha del archivo, sólo por su número de versión.

Un número superior se dará como bueno; a números iguales y diferente tamaño no deberíamos recuperar y en el caso de versión inferior recuperar siempre.

SFC deja siempre un archivo *.log (una especie de historial) en c:\windows, sfclog.txt. ¿utilidad? una dll versión 4.0.320

Instalamos una aplicación y la machaca con 4.1.900. Es superior, luego instalamos otra aplicación y la machaca con 4.0.320

El SFC puede recuperar la versión, ¿pero cuál? la 4.0.320. Hemos perdido la 4.1.900.

El historial nos servirá para saber que la que hemos de recuperar (y que instaló la aplicación) es la 4.1.900 y no la 4.0.320.

Y para saber siempre cual es la aplicación que instala las versiones, sólo necesitamos acceder al log y colocarle una línea de comentarios al instalar cualquiera aplicación.

14. Anexo 1: Modos

Windows es un sistema operativo, sólo se apoya en MS-DOS por cuestiones de compatibilidad.

Comparándolo con NT o LinuX, éstos no se apoyan tampoco en MS-DOS, pero leen la BIOS de la máquina. El paso previo siempre es arrancar la BIOS. En este caso Windows 98 añade un paso intermedio, arranca la BIOS, posteriormente el MS-DOS y finalmente él mismo. Es un paso más y así debemos considerarlo... un "paso" más.

NT y LinuX después de ese inicio cargan su propio HAL (Capa de abstracción del Hardware - Hardware Abstraction Layout-) y pasan ya totalmente de la BIOS, no utilizan los recursos de ésta para nada, no se creen los que les indica y estos S.O. vuelven a verificar los recursos y se inician según sus propias indicaciones.

Hoy parece que existen distribuciones de LinuX que si se apoyan en la BIOS, y también W2000, aunque éste último solo cuando se instala, olvidándose después.

Hecha esta pequeña introducción, los modos del procesador son "REAL", "PROTEGIDO" y "VIRTUAL 8086".

14.1. *Modo Real Y Modo Protegido*

Repasemos con detalle ambos términos antes de la carga de Windows, ya que Windows lo primero que realiza es poner al procesador en modo protegido.

El procesador siempre arranca en modo real (antes de Windows) y las características fundamentales podríamos esquematizarlas de la siguiente forma:

- Restricción de la memoria, sólo se direccionan 20 líneas ($2^{20} = 1$ MEGA).
- Algunas zonas se usan por el hardware, especialmente vídeo o gráfica (posición A000 a C000).
- Desde las posiciones 0 a la 1024 físicas, hay una zona de vectores de interrupción (por software). 256 interrupciones a 4 bytes, 2 para el segmento y 2 para el desplazamiento. Aquí se contienen las direcciones de las rutinas a las que saltará la CPU cuando se haga mediante una INT xx (xx= número) de forma automática. Muchas de estas interrupciones software son diseñadas por la BIOS y otras lo hacen por hardware (IRQ). El manejo lo hace el procesador y está implementado por hardware. Saltará a la correspondiente dirección de la tabla cuando se produce una interrupción y sabe (en modo real) que la tabla empieza en el desplazamiento 0 de la memoria física.

Entonces, **¿que haríamos para que pase a modo protegido?** Parece simple, añadir en unos registros específicos la nueva dirección de la Tabla de interrupciones (ahora excepciones). Se crea una tabla con descriptores de segmento en modo protegido, se le pasa su dirección a registros esenciales del procesador, cambiamos un bit en un registro de control del procesador y hacemos un salto largo. Cuando vuelve del salto, "modo protegido". Ahora ya no parece tan simple.

¿Qué nos ofrece el modo protegido?

El procesador en modo protegido ofrece:

- Control de procesos, protección,
- Protección mediante la memoria virtual y su mecanismo,
- y, virtualización del hardware.

14.2. *Control de procesos. Protección.*

Recordemos que una dirección de memoria es un segmento y un desplazamiento (4 bytes)

Supongamos que antiguamente el segmento(16 bits) ahora es un índice (1, 2,)

y que la dirección real de memoria es lo que consta en una tabla.(Global o Local de direcciones)

Si nuestro segmento contiene un 1 querrá decir que nos estamos refiriendo a la primera posición de la tabla, este contenido se toma igual que antes y sumado al desplazamiento no da la dirección real.

Lo que se llamaba segmento (modo real) y que ahora contiene un índice, pasa a llamarse "descriptor".

Además lo modificamos, aquí puede existir un número desde 0 a 65535, muchas direcciones. Se limita y cambiando los bits utilizados nos quedamos con 8192 posibles elementos en la tabla, suficientes, y que en cualquier momento podemos apuntar los registros que definen la tabla a una nueva, sumando 8192 elementos nuevos.

En realidad la importancia es el uso que le daremos a los bits que dejamos de utilizar, y en particular en dos de ellos, tendremos los valores 0, 1, 2 y 3.

Estos bits en el descriptor nos indicarán el "modo" de funcionamiento del procesador en el segmento de código que a su vez esté definido por el índice del descriptor de la tabla.

El modo "0" es el más bajo y potente. Se realiza todo. Es el modo KERNEL. El código del programa que se ejecute en este modo tiene acceso a todo. El núcleo del S.O. funciona en modo KERNEL.

El modo "1" parece idéntico, pero no puede saltar a modo "0", así como el "2" no puede saltar hacia atrás, mientras que el modo "3" sólo accede a segmentos de uso propio, el menos potente y llamado también modo USER.

En caso de necesitar ejecución de código del sistema operativo y ante la negativa a poder saltar a modo "0", lo hace mediante "excepciones", las cuales están totalmente protegidas, por lo que en teoría (modo protegido) un programa de usuario no podría afectar al sistema operativo o colgarlo.

Gráficamente pintaríamos unos círculos concéntricos, en que el más pequeño sería modo "0" y el más grande el modo "3". Se puede saltar desde el interior al exterior pero no al contrario. Para ello sólo se puede usando las excepciones.

14.3. Memoria Virtual. Su mecanismo.

Antes hemos comentado que una dirección, es un descriptor de segmento y un desplazamiento. Este descriptor de segmento, es un índice que apunta a una tabla y contiene el "segmento" real que sumado al desplazamiento inicial, nos da, la dirección correcta. La dirección es ya la dirección real física de memoria que queremos localizar. (memoria lineal)

De todo este "cisco" no hay que preocuparse. Todo esto lo tiene implementado directamente la CPU. En general esto que estoy describiendo, no es una cosa de la arquitectura 386 (es decir los actuales Pentium). Es algo de "todas" las CPUs. Intel no descubrió nada nuevo con esto (tiene algunas matizaciones...). Todas las CPUs trabajan básicamente con los conceptos descritos.

¿Y sí a este mecanismo de segmentación que nos da una dirección lineal y que en lugar de ser la dirección física, hacemos que apunte a una tabla especial?

La llamaremos "paginación", y aquí cada elemento apuntará a una dirección en memoria real o a una dirección en un archivo de paginación. (swp386 por ejemplo?, o el pagefile.sys en NT??) Y además que se encargue la CPU del trasiego de direcciones.

La memoria la dividimos en páginas de 4Kbs, la dirección de cada una va a una tabla, además añadimos un indicador que marque si es memoria real o está en el archivo de paginación.

El procesador accederá a esa tabla y saltará a la dirección real que se indica que puede ser cualquier otra, o bien, si está en disco, carga la página en una zona libre de la memoria real y le cede el control.

Estamos utilizando una memoria virtual, el programa sólo ve direcciones virtuales que le muestra la tabla de paginación. El procesador se encarga del cambio de páginas de memoria real con las páginas del archivo de paginación y a la inversa.

Los programas ni se enteran. Están en modo virtual. (Bueno la calidad de la memoria virtual no es igual a la RAM, pero todo ayuda)

14.4. Virtualización del hardware

Bien, al igual que se puede poner un mecanismo por el cual las tareas están protegidas, también puede ponerse un bit de marca que nos indique si un descriptor de segmento (es decir el código, real o virtual) al que apunta ese descriptor, está autorizado o no para ejecutar una entrada / salida directa al hardware. Es decir una instrucción IN / OUT en ensamblador.

Si el bit está activo, se lo permite. Si no, lo que hace el procesador cuando encuentra una instrucción de estas, es provocar una "excepción". Esta excepción, tendrá su puerta de tarea y un manejador de esta excepción. Un "manejador" de excepciones, no es nada mas que un "trozo" de código de programa (normalmente perteneciente al sistema operativo, o implementado por algún driver), el cual es el encargado real de ejecutar dicha instrucción o de prohibir su ejecución.

Por eso, por ejemplo, en Windows NT, que tiene virtualización completa del hardware, no podemos realizar ninguna instrucción no permitida, y la mayoría de juegos basados en DOS, que intentan acceder al hardware directamente, son expulsados por la propia CPU. Se dispara una "excepción" y el manejador de dicha "excepción" se lo prohíbe. Recordar que una secuencia mal programada de IN/OUT directos a un puerto, o bien intentando acceder a un puerto inexistente, es lo que provoca las caídas de máquina. Un sistema operativo "serio" no puede dejar que se realicen estas cosas (NT, Linux). Un sistema que quiera guardar compatibilidad con el "viejo" MS-DOS, por desgracia, no virtualiza totalmente el hardware, y deja hacer la mayoría de las cosas a casi todos los programas. Esto es lo que provoca "cuelgues" en Windows 98 por programas mal codificados (o mal "educados").

La pregunta que surge es: **¿si únicamente es un bit, el que "indica" estas protecciones, fácil, por que no lo cambiamos?**

Y la respuesta mas fácil todavía: porque no se puede. No deja la CPU ejecutar una instrucción de este tipo, cuando nuestro programa está en modo USER. únicamente el sistema operativo que está en modo KERNEL es el que puede ejecutar estas instrucciones privilegiadas. Y además si lo intentamos, se provocará otra excepción. Y en esa excepción, el sistema operativo hará lo que considere oportuno: normalmente "matar" a esa tarea.

14.5. *Modo virtual 8086*

Es el tercer modo de funcionamiento del procesador. Simplemente se le da un espacio de direcciones virtuales de un mega y al poner a trabajar el procesador en este modo es como si estuviese en modo real. Pero es "como" si estuviese. No lo está y todas las llamadas podrían ser interceptadas por el sistema operativo.

Pongamos un ejemplo muy claro. Habíamos dicho que la dirección de memoria gráfica para modo texto es el segmento B800. Pues bien, si físicamente "ponemos" en la posición de memoria B800:0000 (una dirección absoluta de memoria), dos bytes, uno de ellos con los atributos de color y otro con la letra "A", veremos instantáneamente la letra "A" con el color seleccionado en la esquina superior izquierda de la pantalla. Es instantáneo, como si la pantalla fuese una "ventana" a esa dirección de memoria.

Esto es en modo real. Pero la pregunta es **¿cómo lo hace Windows cuando abrimos una ventana MSDOS?**. Es importante esto, ya que el escribir en esa posición de memoria por un programa, implicaría que esa "A" saldría instantáneamente en pantalla. Y no sucede esto, lo vemos, eso sí en la parte superior izquierda de una "ventana" MS-DOS que puede estar además físicamente en cualquier posición de nuestro monitor.

Bien, es fácil si hemos entendido un poco el mecanismo de memoria virtual. La solución es "marcar" esas paginas de memoria como "paginadas" en la tabla de paginas. Al ir a utilizar esa dirección, como está teóricamente paginada, ocurre una excepción del procesador, entra a funcionar el mecanismo de excepciones, y un manejador para esta excepción toma el control. Este manejador "se da cuenta" de lo que quería hacer el programa y en vez de hacer lo "normal" que es buscar en el fichero de paginación y traer a memoria física la pagina que falta, lo que hace, es "dibujar" esa "A" que queríamos en la ventana MS-DOS en la esquina superior izquierda.

Parece un poco retorcido..... pero es la única solución, y funciona perfectamente. De cara al programa que se está ejecutando, para él es como si estuviese funcionando en modo real. Es exactamente lo mismo, pero en vez de hacer lo que el quiere, se hace lo que quiere el sistema operativo. además, ese "mega" de memoria, está en cualquier sitio de la memoria física, incluso troceado, da igual. únicamente Windows, construye una tabla de memoria con las paginas que le interese, lo pone en unos registros del procesador y cambia a modo virtual 8086.

14.6. *Multitarea real*

¿Real real? No claro, sólo con más de un procesador podría existir una multitarea real. Pero se le acerca mucho.

El concepto de multitarea lo único que nos incorpora es la posibilidad de repartir el tiempo en unas unidades muy pequeñas, llamadas "quantum", y ceder el control, consecutivamente y por orden de esa cantidad de tiempo asignado a cada tarea de la máquina.

Esto nos muestra "aparentemente" que hay varios programas en ejecución. Realmente, en cada instante del tiempo, (en cada "quantum", dure lo que dure), solo hay UN solo programa en ejecución. Pero en ese quantum, con las velocidades actuales de las CPUs, se pueden hacer muchas cosas.

Evidentemente, aquí intervienen otros dos conceptos:

- 1) Intercambio de tareas.
- 2) Prioridad de las tareas.

El intercambio de tareas, es simplemente el guardar el estado de una tarea cuando ha agotado su tiempo (sus quantums asignados), recuperar el estado de otra tarea y cederle el control. Normalmente esto se puede realizar por el hardware de la CPU (existe una instrucción específica para ello, en "todas" las CPUs del mercado, incluidos los mainframes). Pero Microsoft en vez de utilizar el mecanismo hardware, lo hace por software, "a mano". Esto es muy costoso en ciclos de reloj, pero aparentemente, Microsoft no se fiaba, al menos al principio, de la implementación de este mecanismo en la CPU por parte de Intel.

Prioridad de las tareas, es simplemente un número asignado a cada tarea, que indica el número de "quantums" que puede ejecutar antes de que el sistema operativo tome control y le ceda el control a otra tarea.

Hay que hacer notar, que una tarea, puede "perder" el control, antes de agotar su número de quantums. Sí esa tarea hace una petición de entrada / salida a disco por ejemplo, debido a que esa petición es muy costosa en tiempo (estamos hablando de algunos milisegundos), mientras se ejecuta, evidentemente la CPU puede hacer muchísimas más cosas. Por tanto el sistema operativo toma el control y se lo cede a otra tarea.

El sistema operativo, debe ser "listo". Debe jugar con las prioridades de tarea y variarlas ligeramente. Es decir, a un programa que hace muchas entradas/salida, debido a que está poco tiempo en memoria, el sistema operativo le debería subir la prioridad. En cambio al revés, a un programa que chupa mucha CPU y no realiza apenas entrada / salida, el sistema operativo debe bajarle la prioridad al objeto de que no se apodere todo el tiempo de la CPU.

Evidentemente, el propio núcleo del sistema operativo debe ser el proceso de máxima prioridad.

Bien, esta es la teoría de Multitarea REAL. Y esto es lo que realiza win95 / 98 con las tareas de 32 bits.

Pero las tareas de 16 bits, se "inventaron" (heredado de Windows 3.1), la MULTITAREA CORPORATIVA ("preemptive"). Realmente este no es un concepto informático. Es un concepto Microsoft. Me explico: en este caso no existe la "cesión" de una tarea a otra por el tiempo consumido en máquina, sino como por definición, las tareas Windows, "generalmente" emiten muchos mensajes al sistema, (funcionan por intercambio de mensajes), cada vez que se emiten por parte de un programa ciertos tipos de mensaje, el sistema operativo toma control al recibir ese mensaje y cede en algunos de ellos el control a otra tarea. Este es el funcionamiento de las tareas de 16 bits. Y evidentemente, peligrosísimo, ya que si una tarea no emite esos mensajes, y esa tarea está mal programada y se mete en un bucle infinito, entonces Windows se nos quedará "colgado". No responderá ni el teclado.

Windows 95 / 98, por "herencia", debe permitir ambos tipos de "multitarea". La REAL para tareas de 32 bits, y la "cooperativa" para tareas de 16 bits (y recemos, para que realmente "cooperen" esas tareas). La manera de hacerlo, es crearse una máquina virtual para cada tarea de 32 bits, y otra máquina virtual para "todas" las tareas de 16 bits. Por tanto es importante recalcar que TODAS las de 16, comparten la "misma" máquina virtual.

Esto último implica, que la caída de una tarea de 16 bits, dejará, probablemente, inutilizada TODA la máquina virtual de 16 bits. Y esto es importante, porque win 95 / 98 tiene mucho código de 16 bits en su núcleo.

Por tanto nos inutilizará, probablemente, parte del sistema operativo, obligándonos a reiniciar la máquina.

En cambio, la caída de un programa de 32 bits, únicamente, provocará la caída de esa máquina virtual, y no afecta al resto del entorno Windows. Rearrancando la tarea, esta seguirá en funcionamiento. Por ejemplo, la caída de Word (por lo que sea) no pasa nada, se reanuda Word y sigue funcionando. La caída de una tarea de 16 bits, en el mejor de los casos, lo único que sucedería es que esa tarea ya no podremos arrancarla hasta que reiniciemos la máquina.

15. Anexo 2: El BUG de Windows 98 y SE

Supongo que como yo mismo, mucha gente, habrá sufrido las pantallas azules de Windows 98, y que como vimos en el proceso de inicio de windows98 vienen en su casi totalidad provocadas por los diseños defectuosos de los VxD o controladores de dispositivos y su modo de trabajo, modo real para ser exactos.

Recuerdo que investigué algunos errores que se me hacían inexplicables, la mayoría de las pantallas azules que me sucedía a mí insinuaban que Vmm32.VxD era el causante de los estropicios y en los momentos más inesperados.

Después de revisar con detenimiento las pistas, llegué a una conclusión: No entendía el por qué, y el motivo era para mí simple, los controladores Vmm32 son del propio sistema y muy útiles y necesarios, puesto que su falta o daño podían causar que el sistema no arrancara de ninguna manera.

Los vmm32.vxd son controladores genéricos y normalmente son para sustituir a los específicos en caso de no existir y así poder continuar su funcionamiento (windows98 por supuesto), por ello trabajan en modo real (causa de las imperfecciones, ya que son difícilmente aislables y que el sistema continúe) y digamos que no brillan por su optimización.

Cuando ya había cambiado de S.O. y estaba trabajando con NT y con el incipiente W2000, cayó en mis manos cierta información sobre un BUG (error) en la instalación de Windows, en ese momento mientras leía y releía se me hizo la luz, pensé que era tonto, muy tonto, revisé todo el proceso montones de veces y no caí en darme una vuelta en la instalación de Windows98. Ya a posteriori y no hace demasiado, encontré un artículo referente al tema en una revista del sector (una de las que aparecen los trucos, me parece a mí, Copiados de los grupos de la gente que se los curra y que ni siquiera cita la procedencia), y recuerdo también que algo cité en la lista, pero dejemos el enredo y pasemos a lo técnico.

La instalación de windows98 y 98SE tiene un error no reconocido por Microsoft (que van a reconocer...)

- 1) Cuando Windows inicia busca lo mejor para su funcionamiento, en caso de existir lo utiliza, en caso contrario carga los que están en el directorio Vmm32 (recordemos, VxD en modo real, lentos y poco optimizados).
- 2) Estos controladores son básicamente de sistema, por tanto es el propio Windows quien debe instalar los mejores.
- 3) ¿Que pasa entonces? ¿por qué siempre utiliza los genéricos?

- 4) El BUG!!!! Windows se instala, pero obvia un paso fundamental, NO COPIA los controladores optimizados en el directorio Vmm32, controladores que por otra parte están dentro de los CAB del sistema en el CD.

Solución: Debemos extraerlos manualmente e instalarlos en el directorio Vmm32 de c:\windows\system.

vcomm.vxd, vdmad.vxd, configmg.vxd, vdd.vxd, vmouse.vxd, ntker.vxd y vflatd.vxd.

Se encuentran en los cab 47 y 48 de windows98 y en los 53 y 54 de Windows 98SE.

¿Cómo?: Pues la herramienta SFC del propio sistema operativo.

Inicio Ejecutar SFC

Marcar Extraer un archivo del disco de instalación especificar el archivo

En restaurar de: ruta de los cabs de windows en el cd: ejemplo: D:\win98 (unidad cd)

Guardar archivo en: la ruta es C:\WINDOWS\SYSTEM\MMM32

Se repite para la copia de los archivos mencionados.

Al iniciar Windows, al menos, usará los Vxd que más optimizados están para el propio sistema y que son para cada dispositivo, y no utilizará los genéricos Vmm32.Vxd si no es en caso de corrupción de los primeros.

El sistema funciona con "excepciones", es como ponerse en una cola para saltar del anillo modo USER al anillo modo KERNEL.

Se está leyendo un disquete y a media marcha a propósito o sin quererlo, quitamos el disquete, puede que nos dé un mensaje o puede que nos diga Excepción grave... sobre un fondo azul.

La diferencia estará que la mayoría de las excepciones de este tipo se resuelven volviendo a poner el disquete y pulsar cualquier tecla, mientras que las otras una vez mostrado el azulón no nos va a dejar mojar más pan en la sopa...

Una excepción es una solicitud al sistema para que nos autorice cierta operación con cierto dispositivo, puede que el dispositivo esté ocupado, puede que pille cabreado al procesador, o incluso que pille cabreado al sistema por tanto trabajo y que su respuesta, a falta de formas más sutiles, nos haga entender con ese color tan azulón que está harto de todo, de su trabajo, de su sueldo, de nosotros, de la RAM, de los buses, de los controladores, del registro, vamos que está harto.... y ya sabemos que cuando uno está harto apaga de donde sea...

Agradecimientos

Agradezco enormemente la labor de José Manuel TELLA LLOP (JMT) por su trabajo desinteresado por hacer llegar el conocimiento sobre Windows 98 a todos los usuarios que participaron en los grupos de noticias de Microsoft Windows 98 entre 1998-2000, y al que he tratado de imitar en esta ocasión, congratulándome de conocerlo personalmente.

Gracias a mis amigos Julián Peris&Sve por los ánimos que me dan cada vez que escribo algo.

Al Jefe Paniagua por su apoyo e inclusión en el wwp.

Y en general a todos los componentes de la lista por su paciencia para conmigo.

Bibliografía

- Visión General de Windows
Autor JMT
- Archivos de inicio
Autor JMT
- Modos Real y Protegido
Autor JMT
- Microsoft kits de recursos
MS Corporation 1998
- Notas sobre WindowsMe
Grupos de WindowsMe Betatesters en la fase de testeo del S.O.

Windows 98, el Registro de Windows 98 (c) Microsoft Corporation Rights Reserved