Practicas comunes en sistemas operativos.

Vamos a tratar algunos temas que es necesario conocer a la hora de administrar un sistema informático y que nos van a ser útiles sin importar el sistema operativo concreto con el que trabajaremos. Aprenderemos a realizar acciones sobre el sistema operativo desde el símbolo de comandos, como utilizar los redireccionamientos y las tuberías del sistema, también veremos como crear archivos por lotes (scripts) que nos faciliten las funciones de administración. Estas funciones las estudiaremos usando como referencia los sistemas operativos de Microsoft, aunque la mayoría de los conceptos son muy similares a los usados en otros sistemas operativos.

Estudiaremos además el arranque de un sistema informático típico, la estructura de un sistema de almacenamiento basado en disco duro y haremos un resumen sucinto de la evolución de los sistemas operativos hasta nuestros días.

1 Uso de la línea de comandos.

Normalmente gestionamos los sistemas operativos desde los interfaces gráficos de usuario (IGU) de una forma visual, pero también podemos gestionar dichos sistemas desde la línea de comandos, usando para ello una pantalla de texto plano. La línea de comandos tiene varias ventajas sobre el IGU, como pueden ser:

- Muchas ordenes de gestión del sistema operativo, que se consideran de muy bajo nivel o muy peligrosas, no son accesibles desde el IGU.
- El entorno de texto, es un sistema muy eficiente, podemos abrir sesiones remotas en nuestro equipo desde otras ubicaciones y usar una línea de comandos para dar órdenes al sistema controlado, podemos tener varias sesiones con entorno de texto concurrentes, etc.
- Podemos automatizar las órdenes usando los lenguajes de programación del propio sistema operativo. Estos programas por lotes se conocen como scripts, procesos por lotes o archivos batch y nos ofrecen muchas posibilidades.
- En caso de un error en algún dispositivo hardware del sistema informático, es muy probable que no podamos acceder al IGU, pero casi seguro que será posible acceder de algún modo a la línea de comandos.
- En caso de estar usando herramientas de recuperación de un sistema informático, para intentar corregir un problema de software importante, necesitaremos conocer el uso de la línea de comandos por que seguramente será lo único con lo que contemos.

Normalmente hablamos del intérprete de comandos como un shell. El shell de comandos es un programa de software independiente que proporciona comunicación directa entre el usuario y el sistema operativo. La interfaz de usuario del shell de comandos no es gráfica y proporciona el entorno en que se ejecutan aplicaciones y utilidades basadas en caracteres. El shell de comandos ejecuta programas y muestra su resultado en pantalla mediante caracteres individuales similares al intérprete de comandos de MS-DOS Command.com. El shell de comandos de los sistemas operativos de servidor Windows utiliza el intérprete de comandos Cmd.exe, que carga aplicaciones y dirige el flujo de información entre ellas, para traducir los datos de entrada del usuario a un formato que el sistema operativo reconozca.

CMD no es el único shell de comandos que podemos usar en entornos Windows. Microsoft ha desarrollado otros shell que podemos instalar y usar. Así, tenemos por ejemplo, el MSH que no esta basado en texto sino en objetos y que dispone de muchos más comandos que el CMD. Basado en MSH esta disponible el Nomad, aún en versión Beta, que esta llamado a sustituir al CMD y que presenta potentes opciones de scripting (creación de procesos por lotes) y comandos renovados.

Para ejecutar el shell de comandos de Windows, debemos ejecutar (Tecla Windows + R) el programa CMD.EXE.

1.1 CONFIGURACIÓN DEL SHELL DE COMANDOS.

Para configurar el símbolo del sistema:

1. Abrimos Símbolo del sistema.
2. Hacemos clic en la esquina superior izquierda de la ventana del símbolo del sistema y, a continuación, hacemos clic en Propiedades. (Conseguimos lo mismo si pulsamos Tecla Windows + Barra de espacio)
3. Hacemos clic en la ficha Opciones.

Desde aquí podemos modificar muchas opciones.

- En Historial de comandos, en Tamaño del búfer si escribimos 999 y, a continuación, en Número de búferes escriba o seleccione 5 mejoraremos el tamaño y el comportamiento del buffer de comandos (que nos permite acceder a lo escrito anteriormente con los cursores)
- En Opciones de edición, si activamos las casillas de verificación Modalidad de edición rápida y Modalidad de inserción, conseguiremos habilitar la función de copiar y pegar directamente en el shell de comandos. Para copiar simplemente seleccionamos con el ratón y pulsamos botón derecho del ratón. Para pegar, simplemente pulsamos botón derecho del ratón.
- También podemos modificar el alto y ancho de la pantalla, su posición automática, etc.

1.2 USO DE LA AYUDA EN EL SHELL DE COMANDOS.

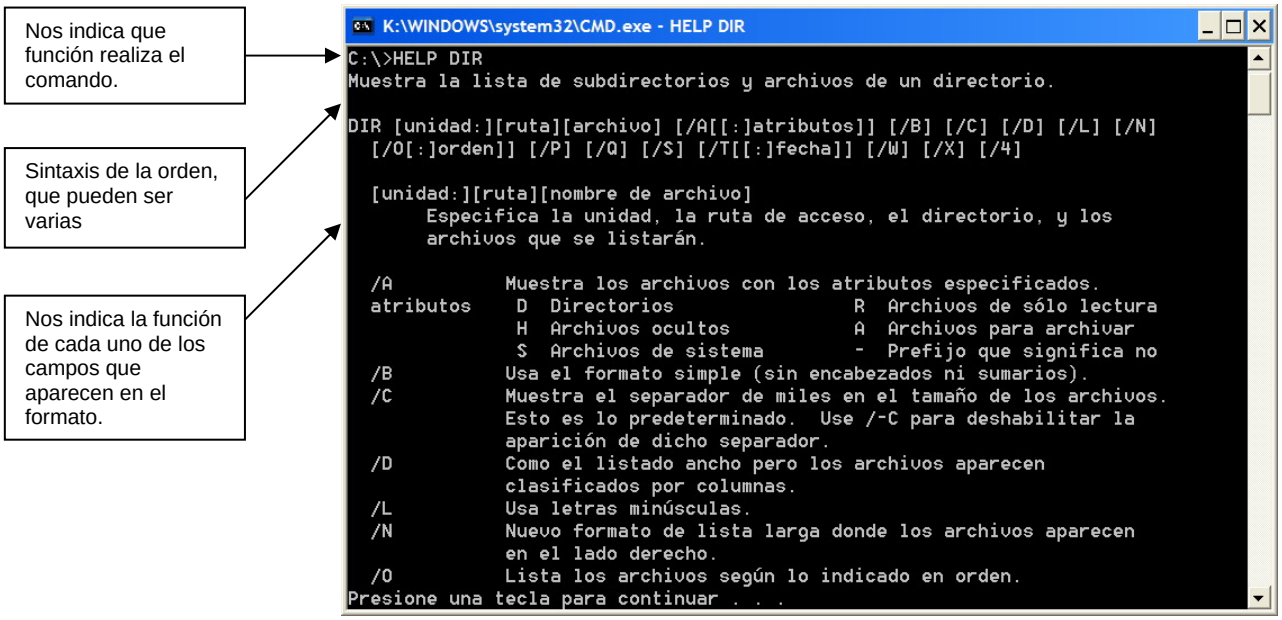
Una de las principales habilidades que debe desarrollar un Administrador de Sistemas, consiste en usar correctamente la ayuda. Cualquier sistema que usemos contará con al menos un nivel de ayuda, que debemos saber buscar e interpretar. En el caso de la línea de comandos, disponemos de una ayuda general accesible mediante la orden HELP. Si queremos ayuda especifica sobre cualquier comando, podemos ejecutar HELP comando. También podemos acceder a la ayuda de un comando escribiendo comando /?.

En caso de que la ayuda que obtengamos con HELP no nos sea suficiente, podemos acceder a la ayuda de Windows XP a la que podemos llegar desde el botón Inicio, Ayuda y Soporte Técnico, donde en Buscar indicaremos el nombre de comando del que deseamos información.

Si tampoco aquí encontramos lo que buscamos, podemos acceder a Internet. Fuentes

importantes son el Google (www.google.com), la Knowledge Base de Microsoft (entrad en www.microsoft.com/spain y allí seleccionad Knowledge Base) y el TechNet de Microsoft (http://www.microsoft.com/spain/technet/)

Es muy importante saber interpretar correctamente las pantallas de ayuda. Existen una serie de convenciones comunes a todos los sistemas que debemos conocer.



La sintaxis aparece en el orden en que debe escribir un comando y los parámetros que lo siguen. La tabla siguiente explica cómo interpretar los diferentes formatos de texto.

Leyenda de formato

Formato	Significado
Cursiva o minúsculas	Información que debe suministrar el usuario
Negrita o mayúsculas	Elementos que debe escribir el usuario exactamente como se muestran
Puntos suspensivos (...)	Parámetro que se puede repetir varias veces en una línea de comandos
Entre corchetes [ ]	Elementos opcionales, pueden usarse o no.
Entre llaves {} opciones separadas por barras verticales |.	Conjunto de opciones de las que el usuario debe elegir sólo una. Ejemplo: {par | impar}

Vamos a insistir en lo que se ha explicado, para asegurarnos de que se entiende bien.

```
DIR [unidad:][ruta][archivo] [/A[[:]atributos]] [/B] [/C] [/D] [/L] [/N]
  [/O[:]orden]] [/P] [/Q] [/S] [/T[[:]fecha]] [/W] [/X] [/4]
```

Veamos que información obtenemos de esta línea, y que significan los caracteres que ahí aparecen.

Las palabras que aparecen sin estar encerradas entre corchetes son palabras obligatorias al formato, es decir que no podemos escribir la orden sin usarlas. Si nos fijamos, solo la palabra DIR esta libre, así que el formato mínimo de la orden seria DIR.

Todo lo que esta encerrado entre corchetes indica que es optativo. Así por ejemplo, el modificador /A es optativo, pero veamos como está representado dicho modificador:

```
[/A[[:]atributos]]
```

Vemos que ahí varios niveles de integración de corchetes. Así, /A es optativo (está entre corchetes) y podemos poner /A sin poner nada más. Podemos poner también /A atributos si queremos, sin poner el símbolo : .Si lo deseamos podemos poner el formato completo que seria /A:atributos.

Lo que se consigue con /A o lo que significan atributos, lo tenemos en la misma ayuda de DIR un poco más abajo.

```
/A          Muestra los archivos con los atributos especificados.
atributos    D  Directorios                R  Archivos de sólo lectura
             H  Archivos ocultos           A  Archivos para archivar
             S  Archivos de sistema        -  Prefijo que significa no
```

Vemos aquí como /A nos sirve para mostrar archivos que cumplan con un determinado atributo. Y vemos como donde en la línea de formato pone atributos, debemos poner una de las siguientes letras: D R H A S. Vemos que también podemos poner el símbolo menos -, pero en este caso se nos indica que es un prefijo, por lo que podríamos poner –A, –S, etc.

Si aprendemos a usar correctamente una pantalla de ayuda, entender lo que esta escrito en ella y lo que se nos quiere decir, habremos dado un paso de gigante para lograr ser Administradores de Sistemas.

1.3 UTILIZAR VARIOS COMANDOS Y SÍMBOLOS DE PROCESAMIENTO CONDICIONAL

Podemos ejecutar varios comandos desde una línea de comandos o secuencia de comandos si utilizamos símbolos de procesamiento condicional. Al ejecutar varios comandos con símbolos de procesamiento condicional, los comandos que hay a la derecha del símbolo de procesamiento condicional actúan basándose en el resultado del comando que hay a la izquierda del símbolo de procesamiento condicional. Por ejemplo, podemos ejecutar un

comando solamente si el anterior causa un error. También podemos ejecutar un comando solamente si el anterior es correcto.

Podemos usar los caracteres especiales enumerados en la tabla siguiente para pasar varios comandos.

Carácter	Sintaxis	Definición
&	Comando1 & Comando2	CMD ejecuta el primer comando, y luego el segundo.
&&	Comando1 && Comando2	CMD ejecuta el primer comando, y si ese comando es correcto, entonces ejecuta el segundo. Si Comando1 falla, no se ejecuta Comando2.
||	Comando1 || Comando2	Comando2 solo se ejecuta si Comando1 es incorrecto o falla.
( )	(Comandos)	Se usa para anidar comandos. Se ejecutan primero los comandos que están dentro de los paréntesis que los que están fuera de los mismos)

1.4 USO DE COMODINES.

Los comodines son caracteres del teclado como el asterisco (*) o el signo de interrogación (?) que se pueden utilizar para representar uno o más caracteres reales al buscar archivos o carpetas. A menudo, los comodines se utilizan en lugar de uno o varios caracteres cuando no se sabe el carácter real o no se desea escribir el nombre completo.

Asterisco (*)

Podemos utilizar el asterisco como sustituto de cero o más caracteres. Si buscamos un archivo que sabemos que comienza por "glos" pero no recordamos el resto del nombre del archivo, escribimos lo siguiente:

glos*

Con esto, buscaremos todos los archivos de cualquier tipo que comiencen por "glos", incluidos Glosario.txt, Glosario.doc y Glos.doc. Para limitar la búsqueda a un tipo de archivo específico, escribimos:

glos*.doc

En este caso, buscaremos todos los archivos que comiencen por "glos" pero con la extensión .doc, como Glosario.doc y Glos.doc.

Signo de interrogación (?)

Podemos utilizar el signo de interrogación como sustituto de un único carácter en un nombre. Por ejemplo, si escribimos

glos?.doc

Encontraremos los archivos Glosa.doc y Glos1.doc, pero no Glosario.doc.

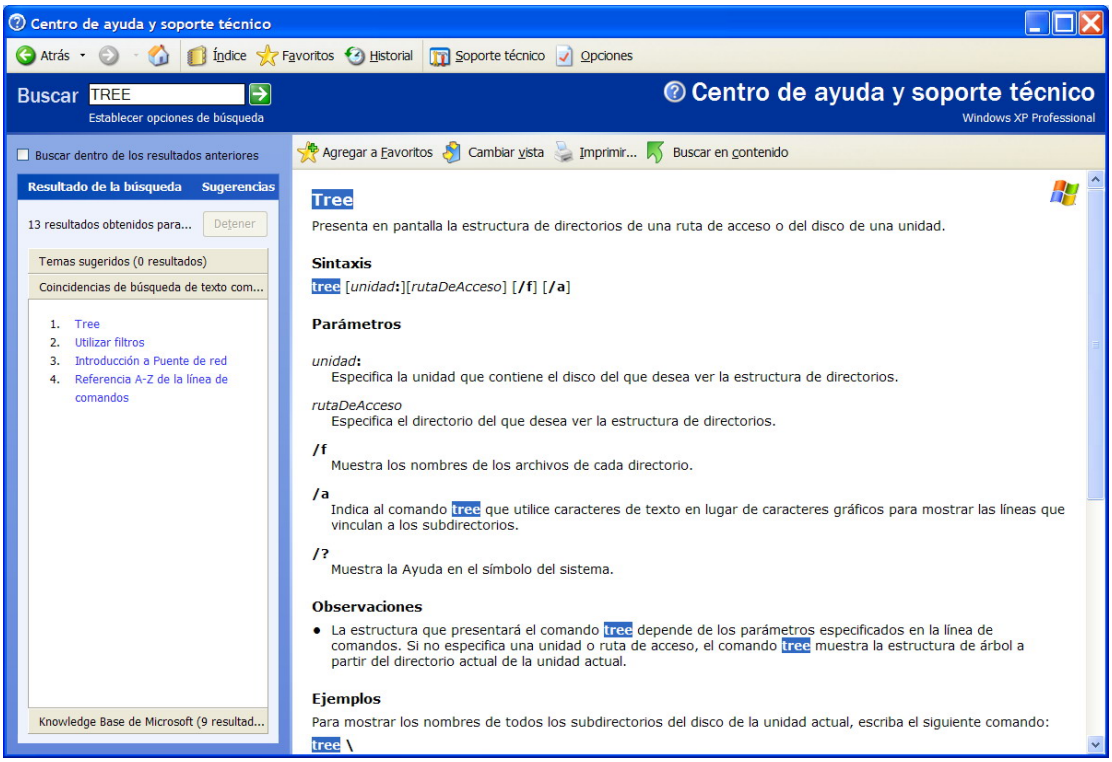
1.5 PRINCIPALES COMANDOS.

En el shell de comandos de Windows, existen cientos de comandos que pueden ser utilizados. Muchos de ellos se instalan directamente con Windows, mientras que otros especiales se instalan conjuntamente con otras herramientas. Veamos los más habituales:

Comando	Descripción	Ejemplo
VER	Muestra la versión del sistema operativo.	VER
Unidad:	Cambia la unidad activa	C: D: E: A:
HELP	Muestra una pequeña ayuda sobre los comandos	HELP HELP comando
DIR	Visualiza el contenido de un directorio	DIR C:\WINDOWS\
ECHO	Muestra mensajes o activa/desactiva el eco local.	ECHO HOLA MUNDO
FORMAT	Formatea una unidad (cuidado)	FORMAT A:
DISKCOPY	Copia un disquete	DISKCOPY A: B:
CHKDSK	Comprueba el estado de un disco	CHKDSK A:
LABEL	Cambia la etiqueta de un disco	LABEL A:
VOL	Muestra la etiqueta de un disco	VOL C:
CLS	Limpia la pantalla	CLS
TIME	Muestra y permite cambiar la hora	TIME
DATE	Muestra y permite cambiar la fecha	DATE
COPY	Permite copiar ficheros	COPY C:\BOOT.INI E:\
MOVE	Mueve ficheros	MOVE C:\BOOT.INI E:\
DEL	Borra ficheros	DEL E:\WINDOWS*.JPG
REN	Renombra ficheros	REN E:\BOOT.INI E:\BT.PUM
MKDIR (MD)	Crea un directorio	MD E:\SACO
RMDIR (RD)	Borra directorios	RD E:\SACO
CHDIR (CD)	Cambia de directorio actual	CD E:\SACO
TREE	Muestra la estructura de directorios	TREE
EDIT	Edita ficheros de texto	EDIT
DOSKEY	Utilidad para recordar líneas de comandos	DOSKEY
EXIT	Sale del símbolo de comandos (si es posible)	EXIT
XCOPY	Copy extendido. Dispone de modificadores exclusivos	XCOPY E:\ D:\ /E
SUBST	Le da un nombre de volumen a un directorio	SUBST J: E:\UTILES

De cada uno de estos comandos podemos obtener ayuda, bien escribiendo HELP comando o escribiendo comando /?

Si esta ayuda no nos es suficiente, podemos acceder al centro de ayuda y soporte técnico de nuestro Windows XP (Inicio – Ayuda y Soporte Técnico) y escribir el comando en el formulario de búsqueda.



Existen muchos más comandos, tanto internos como externos. (Se dice que un comando es interno cuando viene incluido en el propio CMD y se carga en memoria continuamente). Al menos los comandos que aparecen en la tabla de la página anterior, deben conocerse y usarse sin problemas.

No es objetivo de estos apuntes indicar todas y cada una de las ordenes, ni los modificadores posibles en todas ellas. El uso de la ayuda del sistema debería bastar para este fin. Indicaremos ahora algunas órdenes y algunos modificadores a nivel de ejemplo solamente.

- Ejemplo: Visualizar el contenido de un disco. Escribe DIR y Pulsa Intro

Aparecerá un listado de archivos y carpetas, que contienen archivos en su interior, tamaño expresado en bytes, fecha, hora de última actualización (o edición), de la unidad a la que le hemos hecho el DIR. Si el listado es muy largo (hay muchos archivos), veremos como la pantalla va muy rápida y no nos da tiempo a leerlo todo. Para remediar esto escribe el DIR seguido de /P.

- Ejemplo: Listar archivos haciendo pausa. Escribe DIR /P

Una vez la pantalla quede llena, os pedirá que pulséis cualquier tecla para continuar, y así hasta terminar listando todos los archivos y carpetas del disco. También hay otra forma de presentar los archivos por pantalla, visualizándolos a lo ancho.

- Ejemplo: Listar archivos a lo ancho. Escribe DIR /W

Si no cabe en la pantalla, pasará lo mismo que en los casos anteriores, pero esto tiene solución si hacemos servir la función de pausa.

- Ejemplo: Listar archivos a lo ancho con pausa Escribe DIR /W /P

Podéis identificar una carpeta si al hacer el DIR veis unos archivos que lleven a su parte derecha, en vez del tamaño, un nombre: <DIR>, esto significa que esta es una carpeta que contiene, posiblemente, más archivos en su interior.

Ordenar y mostrar el contenido de un directorio ordenado por algún tipo de criterio es, sin duda, una buena herramienta de trabajo. Podemos utilizar las anteriores posibilidades haciendo servir, como habéis podido ver en el ejemplo, la barra inclinada, pero también funciona con los dos puntos. Podéis ampliar estas definiciones si escribís un DIR /?.

- Ejemplo: Listar archivos ordenados de menor a mayor tamaño Escribe DIR /O:S

Los directorios son necesarios para una mejor organización de los discos. Fijaos que el disco duro de vuestro ordenador, al hacer un DIR hay, seguro, una carpeta con el nombre DOS (o un nombre parecido) que contiene todo el sistema operativo DOS.

- Ejemplo: Copiar un fichero de un disco a otro, cambiándole además el nombre.

Escribe COPY C:\FACTURA.TXT A:FACTUR_1.TXT

- Ejemplo: Copiar un fichero a otro disco sin la posibilidad de cambiarle el nombre

Escribe COPY C:\FACTURA.TXT A:

- Ejemplo: Mover un fichero

Escribe MOVE DIBUIX1.BMP A:

- Ejemplo: Mover varios archivos

Escribe MOVE *.BMP A:

- Ejemplo: Borrar un fichero del disco duro

Escribe DEL C:\AMICS.TXT

- Ejemplo: Creación de un directorio para guardar archivos

Escribe MD APUNTS

- Ejemplo: Copiar archivos de un directorio a otro

Escribe COPY A:\ART.TXT C:\APUNTS\HISTORIA

Con la orden EDIT podemos acceder a un editor de textos que viene incluido en todos los sistemas operativos de Microsoft. Este editor tiene la ventaja de trabajar en texto puro, sin incluir símbolos especiales de control como otros editores. Esto es interesante dado que determinados tipos de archivos (procesos por lotes por ejemplo) necesitan estar creados como texto puro, y es más complicado crearlos en otro tipo de editores.

Como ejercicio, probad que todos y cada uno de los comandos de la tabla anterior funcionan, y que comprendemos su función. Aparte, intentad realizar estas acciones con comandos:

1. Crear un directorio en el raíz de nuestro volumen con el nombre TEXTOS. Copiar dentro todos los archivos con extensión TXT que existan dentro del directorio WINDOWS de nuestro volumen.

2. Crear un directorio en el raíz de nuestro volumen con el nombre COPIADO. Moved todos los archivos del directorio TEXTOS creado anteriormente al directorio COPIADO.
3. Crear dentro de COPIADO, un directorio AMIGO y copiad dentro del mismo todos los ficheros con extensión TXT que estén dentro del directorio Windows, incluidos los que puedan estar dentro de subdirectorios.
4. Cread un directorio en el raíz de nuestro volumen con el nombre COPIADO2. Copiad todos los ficheros y directorios de COPIADO a COPIADO2. Atención, queremos que se copien tanto los ficheros como los directorios y sus contenidos.

1.6 REDIRECCIONAMIENTOS Y TUBERÍAS.

Cualquier software que ejecutemos en nuestro sistema informático, va a procesar una información que le llega desde una ENTRADA y va a enviar el resultado del proceso a una SALIDA. Si no indicamos nada, se supone que la entrada será desde el dispositivo por defecto de entrada (stdin) y la salida será al dispositivo por defecto de salida (stdout).

Normalmente en nuestros sistemas, stdin y stdout se refieren a la consola (a la que se referencia en entornos Windows como CON) que esta formada por el teclado como stdin y por el monitor como stdout. Normalmente, además de stdout, nos encontraremos con otra salida que se llama stderror. Mientras por stdout salen los mensajes de salida normales, por stderror salen los mensajes de salida de error.

Con los redireccionamientos, podemos indicar a las órdenes que entrada, salida y salida de errores deben usar, evitando que usen las Standard. Estos redireccionamientos son los siguientes:

>	Redirecciona stdout. Es decir, nos permite indicar una salida para la orden que no sea CON (monitor).
2>	Redirecciona stderror. Es decir, nos permite indicar una salida para los errores de la orden que no sea CON (monitor).
<	Redirecciona stdin. Es decir, nos permite indicar una entrada para la orden que no sea CON (teclado).
>>	Igual que >, pero la salida de la orden se **añade** a la salida que indiquemos. Con > la salida de la orden **reescribe** la salida que indiquemos.
|	El indicador de tubería. Nos permite indicar que la entrada de una **orden** será la salida de otra **orden**. Es decir, el stdout de la 1ª orden, será el stdin de la 2ª orden.

Veamos algunos ejemplos de estas redirecciones y tuberías. Si escribimos `DIR` veremos como esta orden no nos pide nada (no usa stdin) y nos muestra unas líneas (stdout) por pantalla. Vamos a cambiarle stdout, para ello escribimos `DIR > PATATA`. Veremos como por pantalla no nos sale nada, ya que hemos cambiado stdout. Si ahora miramos en el directorio, comprobaremos que se ha creado un fichero PATATA que en su interior (`TYPE PATATA`) contiene la salida de la anterior orden DIR.

¿Qué ocurriría si escribimos las siguientes órdenes?

```
ECHO "HOLA MUNDO" > FICHERO1
ECHO "ESTO ES UN EJEMPLO" > FICHERO1
```

Si ahora miramos el contenido de FICHERO1 veremos como solo contiene la ultima línea. Esto es asi porque > siempre sobrescribe la salida. Para evitar esto escribimos:

```
ECHO "HOLA MUNDO" > FICHERO1
ECHO "ESTO ES UN EJEMPLO" >> FICHERO1
```

Veamos como funciona la redirección de stdin. Si escribimos la orden `TIME` veremos que esta orden si usa stdin, en concreto nos pide que por teclado introduzcamos la hora en formato HH:MM:SS y pulsemos INTRO para cambiar la hora. Bien, escribamos ahora lo siguiente:

```
ECHO 15:00:00 > TIME
```

Si ahora escribimos TIME comprobaremos que ya no nos pide nada, pero que la hora no se ha cambiado. ¿Por qué? Muy simple, estamos enviando la salida de una orden como entrada de otra orden, cosa que no se puede hacer con las redirecciones. Hagamos lo siguiente:

```
EDIT HORA.TXT
```

Nos abrirá el editor de texto con un nuevo fichero que se llama HORA.TXT, dentro de este fichero escribid en la 1ª línea `15:00:00` y en la 2ª línea simplemente pulsad INTRO (dadle entonces a guardar y cerrar). Ahora escribid la siguiente orden:

```
TIME < HORA.TXT
```

Comprobamos como ahora si ha funcionado, la hora se ha cambiado a la deseada.

Veamos ahora la redirección para stderror. Si escribimos

```
MKDIR ONE TWO THREE TWO
```

El sistema creará los tres primeros directorios, pero nos dará un aviso de error, ya que no se ha podido crear el 4º directorio, ya que ya existe.

Escribid ahora

```
MKDIR GUAN TU TRI TU > SALIDA.TXT
```

Veremos como el error sigue apareciendo, ya que hemos redireccionado stdout, pero no stderror. Escribid por fin la línea correcta que seria:

```
MKDIR UNO DOS TRES DOS > SALIDA.TXT 2> ERRORES.TXT
```

Veremos como ahora todo funciona bien. En SALIDA.TXT tendremos la salida normal de la orden, si la hubiera (stdout) y en ERRORES.TXT tendremos la salida de los errores de la orden (stderror).

Usamos la tubería (|) cuando queremos usar la salida de una orden como entrada de la siguiente. Repitamos el ejemplo anterior del echo y el time, pero esta vez con una tubería:

```
ECHO 14:30:00 | TIME
```

Veremos como ahora si funciona perfectamente. Siempre que en una línea queramos usar la salida de una orden como entrada de la siguiente, debemos usar la tubería, no los redireccionamientos.

En todos los sistemas operativos, existen una serie de órdenes especiales conocidas como filtros. Estas órdenes están especialmente diseñadas para trabajar con tuberías, y nos

permiten trabajar con la salida de una orden. Entre las principales que podemos encontrar en los sistemas Windows, tenemos:

SORT	Nos permite ordenar una salida alfabéticamente. Con HELP SORT podemos ver todos sus posibles parámetros.
FIND	Nos permite filtrar una salida, haciendo que solo aparezcan las líneas que contengan una palabra, las que no contengan una palabra, que contemos las líneas que contienen una palabra, etc. HELP FIND.
MORE	Nos permite obtener una salida por pantalla paginada. Es decir, cada vez que la pantalla se llene, nos pide que pulsemos una tecla antes de continuar escribiendo texto.

Así, por ejemplo, si escribimos HELP veremos que nos devuelve una gran cantidad de líneas, posiblemente más de las que seremos capaces de ver por pantalla. En este caso podemos escribir HELP | MORE para paginar la información.

Como ejemplo, cread un fichero tabulado con nombre FAVORITOS.TXT y escribid en el por ejemplo los nombres de varias páginas Web, sus direcciones y su temática, con este formato:

```
El rellano        www.elrellano.com       chistes
El País           www.elpais.es           periódicos
```

……

Si ahora escribimos `TYPE FAVORITOS.TXT | SORT` veremos como obtenemos la lista ordenada desde la primera columna, asi que se ordenará por el nombre de la página.

Si escribimos `TYPE FAVORITOS.TXT | FIND "chistes"`

Veremos como solo nos muestra las líneas donde aparezca la palabra chistes, con lo que es muy fácil filtrar el archivo.

Una línea como la siguiente

```
TYPE FAVORITOS.TXT | FIND "periódicos" > PRENSA.TXT
```

Nos crearía un fichero con nombre PRENSA.TXT que contendría todas las líneas de FAVORITOS.TXT donde aparezca la palabra periódicos.

Como ejercicio, intentad mostrar este fichero por pantalla ordenado por nombre de la página, luego sacadlo ordenado por la dirección, y por ultimo sacadlo ordenado por la temática.

Estos filtros son muy útiles para realizar labores de Administración. Por ejemplo, en http://www.iesromerovargas.net/OASIS/SIM/Documentos/ tenemos un fichero con nombre access.log que contiene los mensajes de información (logs) que el servidor Web del Instituto ha generado en los últimos días. En este fichero tenemos una línea por cada conexión que se ha realizado con el servidor donde se indican que han hecho, de donde vienen, que IP tiene, etc.

Descargar este fichero access.log a vuestro disco duro, y realizar sobre el las siguientes acciones mediante líneas de comandos.

1) Ver que personas se han descargado el TEMA05-01 del servidor.
2) Obtener el número de entradas en el foro de OASIS.
3) Obtener las personas que hayan accedido a la página desde búsquedas en el google. Cread un fichero PREGUNTAS.TXT donde se almacenen las preguntas que han realizado en el google para llegar a nuestra página.

1.7 VARIABLES DE ENTORNO.

El sistema cuenta con sus propias variables, que toman valor cuando se inicia el Sistema. Si queremos ver dichas variables podemos usar la orden SET, que nos muestra una lista de variables ya definidas. Podemos definir nuestras propias variables sin ningún tipo de problemas, basta con poner SET nombre_de_variable = valor.

Es importante no dejar espacios ni delante ni detrás del símbolo =. Asi por ejemplo SET EDAD=18 crea una variable con nombre EDAD y valor 18.

Si queremos acceder al contenido de la variable, encerramos dicha variable entre símbolos de %.

Ejemplo:

```
SET NACIONALIDAD=”Español”
ECHO %NACIONALIDAD%
```

Las variables de entorno tipicas de un sistema Windows, son las siguientes:

Devuelve la ubicación en que las aplicaciones guardan los datos de forma predeterminada.

Variable	Tipo	Descripción
%ALLUSERSPROFILE%	Local	Devuelve la ubicación de perfil Todos los usuarios.
%APPDATA%	Local	Devuelve la ubicación en que las aplicaciones guardan los datos de forma predeterminada.
%CD%	Local	Devuelve la cadena del directorio actual.
%CMDCMDLINE%	Local	Devuelve la línea de comandos exacta utilizada para iniciar el Cmd.exe actual.
%CMDEXTVERSION%	Sistema	Devuelve el número de versión de Extensiones del procesador de comandos actual.
%COMPUTERNAME%	Sistema	Devuelve el nombre del equipo.
%COMSPEC%	Sistema	Devuelve la ruta de acceso exacta al ejecutable del shell de comandos.
%DATE%	Sistema	Devuelve la fecha actual. Utiliza el mismo formato que el comando date /t. Generado por Cdm.exe. Para obtener más información acerca del comando date, vea Fecha.

%ERRORLEVEL%	Sistema	Devuelve el código de error del último comando utilizado. Usualmente, los valores distintos de cero indican que se ha producido un error.
%HOMEDRIVE%	Sistema	Devuelve la letra de unidad de la estación de trabajo local del usuario conectada al directorio principal del usuario.
%HOMEPATH%	Sistema	Devuelve la ruta de acceso completa del directorio principal del usuario. Se establece según el valor del directorio principal. El directorio principal del usuario se especifica en Usuarios y grupos locales.
%HOMESHARE%	Sistema	Devuelve la ruta de acceso de red del directorio principal compartido del usuario. Se establece según el valor del directorio principal. El directorio principal del usuario se especifica en Usuarios y grupos locales.
%LOGONSERVER%	Local	Devuelve el nombre del controlador de dominio que validó la sesión actual.
%NUMBER_OF_PROCESSORS%	Sistema	Especifica el número de procesadores instalados en el equipo.
%OS%	Sistema	Devuelve el nombre del sistema operativo. En Windows 2000 se muestra el sistema operativo Windows NT.
%PATH%	Sistema	Especifica la ruta de acceso de búsqueda para los archivos ejecutables.
%PATHEXT%	Sistema	Devuelve una lista de extensiones de archivo que el sistema operativo considera como ejecutables.
%PROCESSOR_ARCHITECTURE%	Sistema	Devuelve la arquitectura de chip del procesador. Valores: x86 o IA64 (basado en Itanium).
%PROCESSOR_IDENTIFIER%	Sistema	Devuelve una descripción del procesador.
%PROCESSOR_LEVEL%	Sistema	Devuelve el número de modelo del procesador instalados en el equipo.
%PROCESSOR_REVISION%	Sistema	Devuelve el número de revisión del procesador.
%PROMPT%	Local	Devuelve la configuración del símbolo del sistema del intérprete actual. Generado por Cmd.exe.
%RANDOM%	Sistema	Devuelve un número decimal aleatorio entre 0 y 32767. Generado por Cmd.exe.
%SYSTEMDRIVE%	Sistema	Devuelve la unidad que contiene el directorio raíz del sistema operativo de servidor de Windows (es decir, la raíz del sistema).
%SYSTEMROOT%	Sistema	Devuelve la ubicación del directorio del sistema operativo de servidor de Windows.

%TEMP% y %TMP%	Sistema y usuario	Devuelve los directorios temporales predeterminados que utilizan las aplicaciones disponibles para los usuarios conectados actualmente. Algunas aplicaciones requieren TEMP y otras requieren TMP.
%TIME%	Sistema	Devuelve la hora actual. Utiliza el mismo formato que el comando time /t. Generado por Cdm.exe. Para obtener más información acerca del comando time, vea Time.
%USERDOMAIN%	Local	Devuelve el nombre del dominio que contiene la cuenta de usuario.
%USERNAME%	Local	Devuelve el nombre del usuario que ha iniciado la sesión actual.
%USERPROFILE%	Local	Devuelve la ubicación del perfil del usuario actual.
%WINDIR%	Sistema	Devuelve la ubicación del directorio del sistema operativo.

Algunas de estas variables son especialmente importantes, ya que se nos permiten automatizar muchos procesos de Administración. Por ejemplo, si tenemos que ir al directorio Windows para retocar algunos ficheros y en nuestro servidor disponemos de varios sistemas operativos y varios volúmenes de datos, podemos perder mucho tiempo en buscar donde esta situado. Pues un simple `CD %WINDIR%` nos llevaría al directorio de Windows sin posibilidad de error.

Otra variable que usaremos profusamente cuando lleguemos al tema de Windows Server será la de %USERNAME%.

¿Como pequeño ejercicio, como podríamos obtener mediante la orden ECHO por pantalla una línea como la siguiente?

```
Hola, usuario JOSE. Ahora mismo son las 13:17:06,45 del día 09/11/2005
y su directorio actual es M:\Documents and Settings\Jose
```

1.8 PROCESOS POR LOTES. FICHEROS BAT.

Un proceso por lotes es un archivo de texto formado por varios comandos del shell de comandos. Esta secuencia de comandos se ejecuta de uno en uno, línea a línea, en el mismo orden en que aparecen en el programa, como si se tecleara cada uno delante del prompt del sistema (el indicador que nos aparece, como C:\>).

En algunas ocasiones el usuario escribe repetidamente la misma secuencia de comandos para realizar algunas tareas comunes. Para evitar eso podemos colocar esta secuencia de comandos de un archivo de procesamiento por lotes y ejecutar automáticamente todas esa secuencia de comandos.

Para que puedan ser reconocidos por el sistema Windows como archivos especiales, los archivos de procesamiento por lotes deberán llevar la extensión .BAT.

Podemos escribir nuestros propios archivos de proceso por lotes directamente con COPY CON (copy con nombre_fichero, escribimos las líneas y acabamos con Control Z) o bien con EDIT. Recordemos que estos archivos deben ser de texto puro, es decir no pueden ser ficheros de tipo Word o similares.

Estos archivos por lotes, también conocidos como ficheros BAT o scripts, son una de las principales herramientas que usa un Administrador. Nos permiten realizar operaciones tediosas de forma muy rápida, y cuando se aprenden a programar correctamente son tremendamente poderosos.

GESTION DE PARAMETROS EN LOS PROCESOS POR LOTES.

Los parámetros son informaciones adicionales colocadas detrás del nombre de una orden. Si la mayoría de las órdenes del sistema admiten parámetros o modificadores, también será posible gestionar parámetros en los ficheros por lotes.

Vamos a confeccionar un fichero por lotes que borre dos ficheros introducidos como parámetros.

```
@echo off
rem Programa: BORRA2.BAT
del %1
del %2
```

Si ejecutamos este proceso por lotes o script, de la siguiente manera:

```
C:\> borra2 juan.txt maria.txt
```

Esas líneas que hemos creado como `del %1` y del `%2` se transformarían al ejecutarse en `del juan.txt` y `del maria.txt` En la línea de órdenes, cada parámetro debe estar separado con un espacio en blanco. De la forma anteriormente explicada podemos gestionar hasta nueve parámetros (del %1 al %9).

El siguiente ejemplo copia los ficheros introducidos como parámetros al disquete de la unidad B:

```
@echo off
rem Programa: COPIAB.BAT
echo Se están copiando los ficheros %1, %2 y %3 a la unidad B:
copy %1 b:\
copy %2 b:\
copy %3 b:\
```

Ejecución:

```
A:\>copiab juan.bak alberto.bmp marta.dbf
```

No preocuparos de las líneas @echo off y rem, ya que las explicaremos a continuación.

COMANDOS ESPECIFICOS PARA PROCESOS POR LOTES.

Aparte de los comandos ya conocidos, en los procesos por lotes podemos usar algunos comandos especiales, que nos permiten programar estructuras complejas usando procesos. Estos comandos son:

COMANDO CALL (llamar)

Función: Llama un archivo de procesamiento por lotes desde otro igual, sin salir del archivo que hizo la llamada. Es decir, invoca la ejecución de otros archivos como una subrutina o una función.

Formato: `CALL nombre del proceso por lotes`

Una vez terminada la ejecución del proceso por lotes llamado con CALL, se reanuda la ejecución del archivo que hizo la llamada.

Un archivo de proceso por lotes, puede hacer una llamada repetitiva a si mismo (recursividad), siempre y cuando existe una condición de finalización

COMANDO ECHO (eco)

Función: Este comando activa y desactiva la exhibición de comandos por pantalla, o escribe un mensaje por pantalla.

Formato: `ECHO [ON] [OFF] [MENSAJE]`

`Echo ON`: Nos permite ver en pantalla los comandos del DOS que están siendo ejecutados desde un archivo de proceso por lotes

`Echo OFF`: Desactiva la visualización en pantalla de los comandos

El valor por omisión es `ECHO ON`.

`Echo mensaje`: Este comando nos permite imprimir mensajes en la pantalla.

Si escribimos simplemente ECHO, se nos presentará en pantalla el estado actual de ECHO.

Podemos insertar el símbolo @ (arroba), antes de una línea de comandos en un archivo de proceso por lotes, para que no se haga ECO de dicha línea.

Si queremos dejar en pantalla una línea en blanco, se usa ECHO. (ECHO y un punto sin dejar espacios en blanco entre la O y el punto).

COMANDO GOTO (Transferencia control a)

Función; Transfiere el control de proceso a una línea con etiqueta, dentro del archivo de proceso por lotes.

Formato: `GOTO [:] etiqueta`

Etiqueta puede ser cualquier palabra que deseemos.

A continuación se presenta un ejemplo:

```
:INICIO (etiqueta, se reconoce por que empieza por : )
comando-1
```

```
comando-2
comando-3
GOTO :INICIO (saltamos a :INICIO)
comando-4
```

Al encontrarse nuestro programa un comando GOTO, se buscará la etiqueta en el archivo por lotes, si se encuentra, nuestro programa saltará a dicha etiqueta y continuara en la línea que sigue a la etiqueta. En este caso el proceso continuamente ejecuta los comandos 1, 2 y 3, sin llegar nunca al comando 4. En la etiqueta, solo son significativos los 8 primeros caracteres.

COMANDO PAUSE (pausa):

Función: Suspende la ejecución de un archivo de procesamiento por lotes

Sintaxis: `PAUSE (comentario)`

Cuando se esta ejecutando un archivo de procesamiento por lotes, puede ser necesario cambiar el disco o realizar alguna otra operación, por lo cual debemos pausar el proceso hasta que el usuario pulse una tecla.

El comando PAUSE suspende este proceso temporalmente, hasta que se presione cualquier tecla, al ejecutarse emite el siguiente mensaje:

```
Pulse cualquier tecla cuando este listo(a)...
```

Pause comentario: Es útil cuando desea poner en pantalla un mensaje especial. A menos que el ECHO este desactivado PAUSE muestra este "comentario" antes del mensaje oprima una tecla...".

COMANDO REM

Función: Nos permite poner comentarios en el programa.

Formato: `REM comentario`

COMANDO IF (si)

Desvía condicionalmente el proceso de ejecución de un fichero por lotes.

Formatos:

```
IF [NOT] ERRORLEVEL número comando
IF [NOT] cadena1==cadena2 comando
IF [NOT] EXIST archivo comando
```

`NOT` Especifica que Windows XP debe llevar a cabo el comando sólo si la condición es falsa.

`ERRORLEVEL número` Especifica una condición verdadera si el último programa que se ejecutó devolvió un código de salida igual o mayor que el número especificado.

`cadena1==cadena2` Especifica una condición verdadera si las cadenas de texto especificadas coinciden.

`EXIST archivo` Especifica una condición verdadera si el archivo especificado existe.

comando Especifica el comando que se ejecutará si se cumple la condición. Comando puede ir seguido de la palabra clave ELSE y, a continuación, un comando que se ejecutará si la condición especificada es FALSA

La cláusula ELSE debe aparecer en la misma línea que la del comando que sigue al IF Por ejemplo:

```
IF EXIST archivo (
    del archivo
) ELSE (
    echo archivo no existente.
)
```

Lo siguiente NO funcionará porque el comando DEL debe terminarse con una nueva línea o estar entre paréntesis:

IF *EXIST archivo del archivo ELSE echo archivo no existente*

Tampoco funcionará lo siguiente, ya que el comando ELSE debe estar en la misma línea del comando IF a menos que se usen paréntesis:

IF EXIST archivo del archivo.

ELSE echo archivo no existente

Si deseamos tenerlo todo en una misma línea, lo siguiente funcionaría:

```
IF EXIST archivo (del archivo) ELSE echo archivo no existente
```

También podemos realizar distintos tipos de comparaciones usando el siguiente formato:

```
IF [/I] cadena1 comparación cadena2 comando
```

donde comparación puede ser:

EQU – igual (equal)

NEQ - no igual

LSS - menor que

LEQ - menor que o igual

GTR - mayor que

GEQ - mayor que o igual

si el modificador /I, si se especifica, realiza comparaciones de cadena que no distinguen entre mayúsculas y minúsculas. El modificador /I también puede usarse en la forma cadena1==cadena2 de IF.

Veamos algunos ejemplos de estructuras IF.

```
@ECHO OFF
IF “23” LSS “12” (
      ECHO pues resulta que 23 es menor que 12
) ELSE (
      ECHO pues resulta que 23 no es menor que 12
)
```

```
-----------------------------------
@ECHO OFF
IF EXIST “C:\BOOT.INI” (
      ECHO “EL FICHERO C:\BOOT.INI EXISTE”
) ELSE
      ECHO “EL FICHERO C:\BOOT.INI NO EXISTE”
)
-----------------------------------
@ECHO OFF
IF %1 EQU %2 (
      ECHO Los dos parámetros pasados son iguales
) ELSE (
      IF %1 LSS %2 (
            ECHO El primer parámetro es menor
      ) ELSE (
            ECHO El segundo parámetro es mayor
      )
)
-----------------------------------
@Echo Off
If %USERNAME% EQU “Jose Antonio” (
      ECHO Esta es tu sistema, Bienvenido.
) ELSE (
      ECHO Y tu quien eres? FUERA DE AQUÍ, HOMBRE!
)
-----------------------------------
@Echo Off
If %DATE% EQU “15/02/2006” ECHO Cumpleaños feliiii
```

Comando FOR (para)

Su formato es:

```
FOR %%variable IN (conjunto) DO orden
```

Esta orden repite la orden especificada para cada valor del conjunto. Conjunto es una lista de datos. En ella, se pueden establecer varios nombres separados por espacios y también utilizar comodines para representar ficheros o directorios.

Ejemplo

```
for %%I in (juan.txt maria.txt cinta.dat) do type %%i
```

La variable %%I va tomando cada uno de los valores del conjunto y se los envía a la orden Type. En este ejemplo se visualizan en pantalla los ficheros JUAN.TXT, MARIA.TXT y CINTA.DAT.

Como se ve en el ejemplo, las variables del FOR no tienen un solo % como los parámetros, ni están encerradas entre % como las variables del sistema, sino que comienzan por %%.

Este otro ejemplo, escribiría por pantalla los días de la semana:

```
For %%n in (lunes, martes, miércoles, jueves, viernes, sábado, domingo) do ECHO %%n
```

Después del do solo podemos poner una orden, aunque podemos usar paréntesis como hacíamos en el IF.

```
For %%N in (Juana, Paco, Jose, Eduardo, Juan, Ango) do (
      If %%N EQU "Juana" (
            Echo Juana es mi única profesora
      ) ELSE ( Echo uno de mis profesores es %%N )
      )
```

También podemos usar el For para trabajar con ficheros:

```
For %%F in (C:\WINDOWS\*.TXT) do (
      Echo Procesando el fichero %%F
      Copy %%F C:\COPIA_SEGURIDAD
      Echo Ya he copiado el fichero %%F
      )
```

Algunos procesos por lotes de ejemplo.

1. BUSCAR.BAT. Un proceso por lotes que acepte como parámetro el nombre de un fichero. Dicho proceso nos mostrará por pantalla los directorios donde dicho fichero exista en el volumen C, si es que existe en alguno.
2. RELOJ.BAT. Un proceso por lotes que muestre por pantalla la hora y la fecha actual, pero ojo, queremos que por pantalla nos muestre SOLO la hora y la fecha, no que nos la pregunte.
3. PROGRAMA.BAT. En muchas ocasiones, cuando un usuario recibe una aplicación desconocida, se ve obligado a buscar el fichero ejecutable que haga funcionar la aplicación. El fichero por lotes PROGRAMA.BAT busca todos los ficheros ejecutables (aquellos con extensión COM, EXE o BAT) del directorio actual y los visualiza en pantalla ORDENADOS por nombre.
4. XDEL2.BAT. Un proceso por lotes que acepte como parámetro un nombre de fichero. El proceso borrará el fichero, pero antes de borrarlo lo copiara al directorio CUBO_BASURA que cuelga del raíz. Hay que comprobar si el directorio CUBO_BASURA existe, y crearlo si no es asi. También hay que comprobar que el fichero pasado como parámetro 1 existe, y si no es así, indicarlo en un mensaje en pantalla.

5. ES_MES.BAT. Un proceso por lotes que acepte un parámetro. Por pantalla debe aparecer el mensaje HA INTRODUCIDO UN MES COMO PARAMETRO o ESO NO ES UN MES RECONOCIDO. Obviamente, obtendremos el primer mensaje cuando el parámetro sea el nombre de un mes en mayúsculas, obtendremos el segundo mensaje cuando lo que el usuario introduzca no sea el nombre de un mes en mayúsculas.
6. ADIVINA.BAT. Un proceso por lotes que acepta como parámetro un nombre de usuario. Debe comprobar si el nombre de usuario es igual al nombre de usuario del sistema. En caso de ser iguales, dirá por pantalla ACCESO CONCEDIDO. Si no coinciden los nombres, por pantalla saldrá el mensaje ACCESO DENEGADO una vez detrás de otra, metiéndose en un bucle sin salida.
7. EL_MAYOR.BAT. Un proceso por lotes que acepte 3 parámetros, que deberán ser 3 números entre el 1 y el 99. Posteriormente el proceso indicará por pantalla: EL NUMERO MAYOR ES y aparecerá el mayor de los 3 números introducidos. No hay que controlar errores.
8. EL_MENOR.BAT. Un proceso por lotes que acepte 9 parametros, que deberán ser 9 números entre el 1 y el 99. Posteriormente el proceso indicará por pantalla: EL NUMERO MENOR ES y aparecerá el menor de los 9 números introducidos. No hay que controlar errores.

Soluciones a los procesos por lotes propuestos.

1. BUSCAR.BAT

```
@ECHO OFF
DIR C:\%1 /S /B
```

2. RELOJ.BAT

```
@ECHO OFF
ECHO %TIME%     -      %DATE%
```

3. PROGRAMA.BAT

```
@ECHO OFF
DIR *.EXE /B > FICHEROS
DIR *.COM /B >> FICHEROS
DIR *.BAT /B >> FICHEROS
TYPE FICHEROS | SORT
```

4. XDEL2.BAT

```
@ECHO OFF
IF NOT EXIST C:\CUBO_BASURA ( MD C:\CUBO_BASURA )
IF k%1k==kk ( GOTO :ERROR )
IF EXIST %1 (
             ECHO El fichero existe. Copiándolo a CUBO
             COPY %1 C:\CUBO_BASURA
             ECHO Eliminando el fichero
             DEL %1
) ELSE (
             ECHO El fichero no existe. No se puede borrar.
)
GOTO :FINAL
:ERROR
ECHO No ha usado el parámetro. Formato de la orden: XDEL fichero
:FINAL
```

5. ES_MES.BAT

```
@ECHO OFF
IF k%1k==kk ( GOTO :ERROR )
FOR %%M IN (ENE,FEB,MAR,ABR,MAY,JUN,JUL,AGO,SEP,OCT,NOV,DIC) DO (
             IF %%M EQU %1 ( GOTO :SI_ES_MES )
)
ECHO Lo que ha introducido no es un mes.
GOTO :FINAL
:SI_ES_MES
Echo Lo que ha introducido como parámetro es un mes.
GOTO :FINAL
:ERROR
ECHO No ha usado el parámetro. Formato de la orden: ES_MES mes
:FINAL
```

6. ADIVINA.BAT

```
@ECHO OFF
IF k%1k==kk ( GOTO :ERROR )
IF %1 EQU %USERNAME% (
      Echo Bienvenido
) ELSE (
      :BUCLE_SIN_FIN
```

```
      ECHO Acceso Denegado!!!!!
      GOTO :BUCLE_SIN_FIN
)
GOTO :FINAL
:ERROR
ECHO Por favor, introduzca un nombre de usuario como parámetro.
:FINAL
```

7. EL MAYOR.BAT

```
@ECHO OFF
IF %1 GTR %2 (
        IF %1 GTR %3 (
              ECHO El mayor es el 1º, que es %1
        ) ELSE (
              ECHO El mayor es el 3º, que es %3
        )
) ELSE (
        IF %2 GTR %3 (
              ECHO El mayor es el 2º, que es %2
        ) ELSE (
              ECHO El mayor es el 3º, que es %3
        )
)

---- OTRA SOLUCION -----
@ECHO OFF
SET MAYOR=%1
IF %2 GTR %MAYOR% ( SET MAYOR=%2 )
IF %3 GTR %MAYOR% ( SET MAYOR=%3 )
ECHO El mayor es %MAYOR%
```

8. EL MENOR.BAT

```
@ECHO OFF
SET MENOR=%1
FOR %%N IN (%2,%3,%4,%5,%6,%7,%8,%9) DO (
      IF %%N LSS %MENOR% ( SET MENOR=%%N )
)
ECHO El menor es %MENOR%
```

OJO. Aunque este programa 8 parece correcto, no lo es. Las variables toman valor en un bucle for ANTES de que este se ejecute, por lo que %MENOR% no irá actualizando su valor, y por lo tanto el programa no funcionará. Colocad un comentario en la línea @ECHO OFF y ejecutad el programa, para verlo más claro.

Podemos resolver esto usando un CALL para crear una función, pero no quiero complicarlo en exceso (en el próximo punto veremos una pista de cómo se resolvería), asi que este programa habría que resolverlo usando la 2ª solución del ejercicio 7, es decir:

```
---- OTRA SOLUCION -----
@ECHO OFF
SET MENOR=%1
IF %2 LSS %MENOR% ( SET MENOR=%2 )
IF %3 LSS %MENOR% ( SET MENOR=%3 )
…….
IF %9 LSS %MENOR% ( SET MENOR=%9 )
ECHO El menor es %MENOR%
```

Comentario sobre CMD y MSH

Como indique al principio de estos apuntes sobre shell, CMD es el shell que se incorpora con Windows por defecto, pero podemos probar otro shell como MSH (Monad).

Veamos un ejemplo.

FACTORIAL.BAT. Realizar un proceso por lotes que obtenga el factorial de 10.

Este proceso podría hacerse en CMD asi (aquí si utilizo funciones y técnicas avanzadas):

```
@ECHO OFF
SET FACT=1
FOR %%N IN (2,3,4,5,6,7,8,9) DO ( CALL :CALCULO "%%N" )
ECHO %FACT%
GOTO :FINAL
:CALCULO
SET /A FACT=%FACT%*%1
:FINAL
```

Si este mismo proceso quisiéramos hacerlo usando MSH (Monad) sería así:

```
$f=1
foreach ($i in 2..10) { $f *= $i }
$f
```

O también podría ser: `for ($i = 10; $i -gt 1; $i--) { $f *= $i }`

2 Arranque de un Sistema Informático.

Ya hemos visto anteriormente que el hardware, por si solo es totalmente incapaz de realizar ninguna acción. Necesita un software que le indique que tiene que hacer. Cuando encendemos un sistema informático, estamos poniendo en marcha hardware, por lo que se necesitan medios especiales para hacer que se cargue un primer software.

En los ordenadores compatibles actuales, el proceso de carga de un sistema operativo por ejemplo DOS, Windows o Linux se compone de una serie de pasos que se inician cuando se conecta o reinicia el ordenador. El proceso comienza siempre en la BIOS, y salvando algunas pequeñas variaciones que puede haber en función de cada fabricante de hardware y de la propia BIOS, el desarrollo paso a paso de esta secuencia es el siguiente:

1. Cuando se da tensión a la fuente de alimentación y una vez que la alimentación se estabiliza, genera una señal "Power Good" en uno de los cables que va de la fuente de alimentación a la placa base; esta señal es recibida en el juego de chips instalado en la referida placa, y a su vez generan una señal de reinicio (reset) al procesador. La finalidad de este proceso es evitar que el procesador arranque prematuramente, cuando las tensiones de alimentación no son todavía correctas, lo que podría producir daños en el hardware. Es el mismo sistema que se utiliza para un reinicio en caliente cuando pulsa en el botón marcado "Reset".

   Nota: Precisamente, debido a este mecanismo, en algunos casos de fuentes de alimentación defectuosas se originan súbitos e imprevistos resets del sistema cuando la tensión baja demasiado y luego se restablece a valores correctos.

2. El procesador arranca cuando se retira la señal de reset. En este momento no existe en su memoria ninguna instrucción o dato, por lo que no puede hacer absolutamente nada. Para salvar el obstáculo, los fabricantes incluyen en la circuitería (hardware) de la placa base un mecanismo especial. El sistema se dirige a una dirección fija de memoria FFFF0h. Esta dirección, situada muy cerca del final de la memoria del sistema en los primeros ordenadores compatibles, es el punto de inicio de la BIOS. En realidad este punto de inicio contiene una instrucción de salto (jump) que indica al procesador donde tiene que dirigirse para encontrar el punto donde comienza realmente el programa de carga (BOOTSTRAP) de la BIOS. Este programa contenido en esa dirección se lleva a la CPU y se ejecuta.

3. La primera parte del programa de la BIOS inicia un proceso de comprobación del hardware denominado POST (Power-On Self Test), en caso de existir errores graves, el programa se detiene emitiendo una serie de pitidos (http://bioscentral.com) que indican el tipo de error encontrado; el orden de las comprobaciones del POST depende del fabricante, pero generalmente la secuencia de comprobaciones se resume como sigue:

   a. Comprobación de registros del procesador
   b. Varias comprobaciones sobre la memoria RAM
   c. Inicializar los dispositivos de video y teclado.
   d. Determinar el tamaño de la RAM completa y comprobar su funcionamiento (el recuento que se ve en pantalla). Si llegado a este punto existiera algún

error en la memoria se mostraría un mensaje de error (el dispositivo de video ya está operativo).

e. Inicializar los puertos: COM (comunicaciones serie), LPT (comunicaciones paralelo), USB, S-ATA, SCSI, etc.

f. Inicializar, en su caso, el sistema de disquete.

g. Inicializar el sistema IDE, S-ATA o SCSI. (Discos duros, CDROMS, etc.).

4. La comprobación del dispositivo de video incluye cargar y ejecuta la parte de BIOS incluida en el adaptador de video. La mayoría de las adaptadoras modernas muestran en pantalla información sobre sí mismas; es por esta razón por la que, a veces, lo primero que se ve en pantalla es información sobre la propia controladora de video antes que ningún mensaje de la BIOS del sistema.

   Nota: Si se trata de un reinicio en caliente ("Hot boot"), que puede conseguirse con la combinación [Ctrl]+[Alt]+[Sup], la fase de comprobación POST se omite, y el proceso de carga sigue desde el siguiente punto.

5. A continuación del POST, la BIOS recorre la memoria en busca de la posible existencia de otros programas en ROM para ver si alguno tiene BIOS, lo que ocurre por ejemplo, con los controladores de disco duro IDE/ATA, cuyas BIOS se encuentran en la dirección C8000h; otros elementos que suelen contar con sus propias BIOS son las tarjetas de red y las controladoras SCSI. Estos módulos son cargados y ejecutados.

6. A continuación, el BIOS muestra su pantalla inicial (generalmente con los créditos del fabricante número de versión y fecha). Como hemos visto, el BIOS realiza una especie de inventario del sistema y algunas pruebas para verificar que su funcionamiento es correcto. En los PCs originales la especificación del hardware disponible se efectuaba mediante interruptores ("Jumpers") situados en la placa-base. A partir de los ATs (80286) se dispone de una memoria permanente, accesible para el usuario (ROM del Sistema), donde está inventariado el hardware básico y su tipo. La tendencia actual es el estándar PnP (Plug and Play). Si la BIOS lo soporta, es capaz por sí misma de detectar y configurar los dispositivos conectados, asignándoles los recursos necesarios y mostrando un mensaje en pantalla por cada uno instalado. Las BIOS modernas pueden detectar automáticamente los parámetros del tipo de disco duro y su forma de acceso. Finalmente, la BIOS muestra en pantalla un resumen de la configuración del sistema. (Podemos pulsar la tecla Pause en este momento para ver tranquilamente la tabla que normalmente aparece en pantalla).

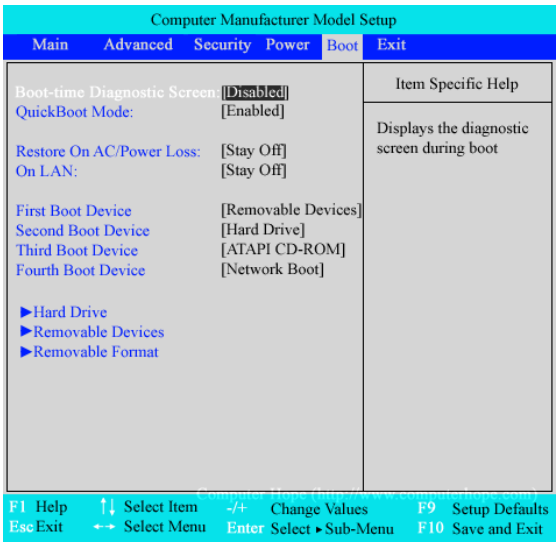
7. Una vez llegado a este punto, el sistema informático ha determinado que todo el hardware del mismo se encuentra en condiciones de funcionamiento, y el pequeño programa que esta almacenado en la ROM de nuestro sistema se acaba. Pero antes de terminar su ejecución, debe "ceder" el control del sistema a otro software. Este software es el sistema operativo. Pero ¿Dónde buscará el sistema operativo a cargar nuestro sistema informático? Y en caso de que existan varios sistemas operativos en varios soportes, ¿cual de ellos será el elegido?

2.1 ELECCIÓN Y ARRANQUE DEL SISTEMA OPERATIVO.

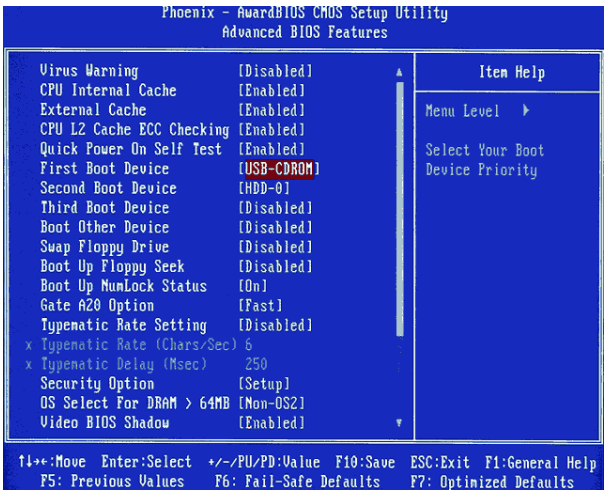
En este punto en el que estamos, el programa que esta en la CPU es el POST, y ya ha concluido todo su trabajo. Pero si dicho programa simplemente liberará la CPU, el equipo se quedaría colgado ya que ningún otro software entraría en el microprocesador. Por ello, la última misión del POST es buscar otro programa, y cargarlo en la CPU antes de liberarla. En un sistema informático actual podemos tener tres discos duros, cada uno de ellos con varias particiones donde pueden estar almacenados varios sistemas operativos; podemos tener un CD en la unidad lectora que también cuente con su propio sistema operativo; podemos tener un disquete de inicio en la disquetera; podemos tener un pequeño sistema operativo en un dispositivo USB; podemos tener un disco duro externo conectado por FireWire; etc. ¿Cómo puede saber el POST a cual de todos estos programas cederle el control?

De momento, en la BIOS de casi todos los equipos modernos es posible encontrar unas opciones que indican cual es el soporte de información desde el cual se va a arrancar el sistema (Boot).

Normalmente estas opciones se encuentran en la segunda opción que aparece en el menú de la BIOS (opciones avanzadas de la BIOS ó Advanced BIOS Features). En alguna opción de este menú, normalmente se nos permite indicar varios dispositivos ordenados que utilizaremos para el arranque. Una opción que se puede dejar por defecto, es indicar que se arranque desde el Floppy, luego desde el CD, y por fin del HDD, para que nos permita arrancar el sistema desde disquete, si no existe desde CD, y si tampoco hay ningún CD de arranque, desde el disco duro. En las BIOS más modernas, veremos que también podemos indicarle que arranque desde un puerto USB, desde un puerto S-ATA, etc.



Vemos aquí algunos de los formatos escogidos por las BIOS de distintos fabricantes para indicar el orden de los dispositivos de arranque.



Si el sistema operativo se ejecuta desde disquete o CD, no hay demasiados problemas, dado que en un disquete o en un CD solo puede haber un único proceso de arranque para un único sistema operativo. Sin embargo, es posible que en disco duro tengamos varios sistemas operativos para arrancar en nuestra maquina en varias particiones. Además, podemos tener hasta 4 discos duros normalmente en nuestro sistema, y en cada disco podemos tener varios sistemas operativos instalados.

Desde la BIOS vemos como podemos indicar de que dispositivo queremos arrancar. Aquí podemos indicar normalmente si queremos arrancar desde el disco duro, desde el CD, USB, etc. Hay BIOS desde donde se puede indicar incluso desde cual de los discos duros queremos arrancar (HDD-0, HDD-1, etc.) Hay que tener en cuenta que en algunas BIOS esta facilidad para distinguir entre los distintos discos duros no esta presente, o no funciona bien. En los casos en que esto ocurra, tendremos que introducirnos en la BIOS y desactivar los discos duros de los que no queremos que arranque. Así, por ejemplo, en un sistema informático de dos discos duros si queremos arrancar desde el primer disco duro no tenemos que hacer nada pero si queremos arrancar desde el segundo disco duro desactivaremos el primero en la BIOS. Para desactivar los discos duros, hay que entrar en la primera opción de la BIOS y poner none, not instaled, o algo parecido en el tipo de disco duro que queremos desactivar. Esto no quiere decir que dichos discos duros no se usarán durante el funcionamiento normal de la maquina, sino que no se usarán en el proceso de arranque.

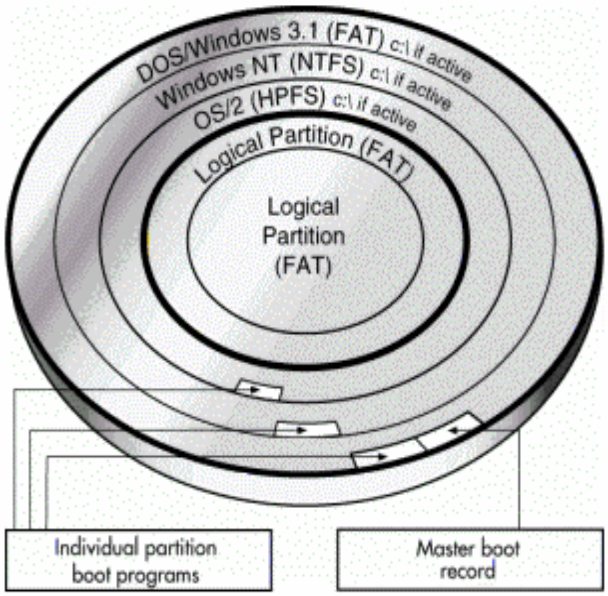
Pero con esto conseguimos indicar al sistema informático que disco duro quiero utilizar para el arranque del sistema… pero resulta que en un solo disco duro puedo tener instalado más de un sistema operativo. ¿Cómo se le indica al sistema que quiero arrancar con Windows XP, o con Linux, o con Beos si todos están instalados en el mismo disco duro? Para entender esto tenemos que comprender bien como esta organizado un disco duro.

2.2 ORGANIZACIÓN LÓGICA DE UN DISCO DURO.

Vamos a ver como organiza el sistema operativo el disco duro. Es aconsejable antes de entrar en este tema, volver a leer el tema referente a los soportes de almacenamiento, en especial los puntos donde se estudiaron los disquetes flexibles y los discos duros, ya que vamos a hacer referencia a varios conceptos que se trataron en dichos puntos.

Los discos duros presentan una serie de diferencias frente a la estructura lógica de los discos flexibles:

1. Tienen una tabla de particiones en el primer sector.
2. Pueden crearse de una a cuatro particiones.
3. Cada partición tiene su propio sector de arranque.



Las particiones son divisiones lógicas efectuadas en un disco duro. Responden a una necesidad muy importante en informática: compartir un mismo disco duro para varios sistemas operativos. Cada partición tiene la estructura lógica correspondiente a su sistema operativo. Una partición dos contiene sector de arranque, FAT, directorio raíz y área de datos, una partición NTFS tiene su sector de arranque

y MFT, etc. Los datos de una partición no se mezclan con los de otra.

En un disco duro podemos tener hasta 4 particiones como máximo. De las 4, solo una puede estar definida como **activa** al mismo tiempo. Esta partición activa será la que cargue el sistema operativo cuando iniciamos el sistema informático. En el primer sector de todo disco duro no se sitúa un sector de arranque (puede haber un sector de arranque por cada partición, por lo que es posible que en un disco duro existan 4 sectores de arranque), en su lugar se sitúa una tabla de particiones (Master Boot Record o MBR). Esta tabla de particiones incluye una tabla donde definimos las 4 particiones que pueden estar presentes en nuestro disco duro y un pequeño programa que permite localizar la partición activa, leer su sector de arranque y usarlo para arrancar nuestro sistema informático.

Esta tabla de particiones (MBR) está situada en el primer sector del disco duro, de modo que su tamaño es de 512 bytes. En esta capacidad se almacena lo siguiente por cada MBR:

Dirección.	Contenido.	Tipo.
+000h	Programa MBR.	445 Bytes.
+1BEh	1º entrada de la tabla de particiones	16 Bytes
+1CEh	2º entrada de la tabla de particiones	16 Bytes
+1DEh	3º entrada de la tabla de particiones	16 Bytes
+1EEh	4º entrada de la tabla de particiones	16 Bytes
+1FEh	Identificación (AA55h)	2 Bytes

Contenido del Master Boot Record o MBR.
Longitud = 200h = 512 Bytes.
El código AA55h marca este sector como ejecutable.

Vemos como existe un programa al principio conocido como programa MBR o gestor de arranque que ocupa 445 Bytes. Un programa MBR estándar, leerá la tabla de particiones y escogerá de cual de esas particiones va a arrancar el sistema operativo. No lo hará como podría parecer lógico de la primera partición, sino de la partición primaria que esta marcada como activa. El MBR lee el primer sector de esa partición, y le cede el control de la CPU a ese programa (Boot Sector).

Hay que indicar que no existe un programa MBR estándar. En realidad, el código que se encuentra aquí, puede ser muy variado, aunque normalmente todos son compatibles. Podemos instalar programas MBR conocidos como gestores de arranque que amplían las posibilidades el gestor de arranque MBR instalado por defecto.

Hay que prestar atención a lo que se ha dicho. Si se arranca desde un disquete, se lee solo el primer sector (Boot Sector). Sin embargo si se arranca desde un disco duro, se lee el primer sector (MBR) y este a su vez, lee un segundo sector (Boot Sector).Vemos también como existen 4 entradas para almacenar hasta 4 particiones. De aquí viene el límite de 4 particiones para un disco duro. Por cada una de estas entradas de 16 Bytes se almacena lo siguiente:

Dirección.	Contenido.	Tipo.
+00h	Estado de la partición: 00h – Inactiva 80h – arranque (activa)	1 Byte
+01h	Cabeza de lectura / escritura donde comienza la partición.	1 Byte
+02h	Sector y cilindro donde comienza la partición.	2 Bytes
+04h	Tipo de partición: 00h – Libre 01h – DOS con la vieja FAT de 12 bits. 02h – XENIX 03h – XENIX 04h – DOS con FAT 16 05h – Partición extendida. 06h – Partición DOS > 32 Megas. 0Bh – Windows FAT32 0Ch – Windows FAT 32 LBA 0Eh – VFAT 16h – Hidden FAT 16 (Oculta) 63h – Unix 65h – Novell Netware Etc.....	1 Byte
+05h	Cabeza de lectura / escritura donde termina la partición.	1 Byte
+06h	Sector y cilindro donde termina la partición.	2 Bytes
+08h	Dirección del primer sector de la partición. (Sector de arranque).	4 Bytes
+0Ch	Número de sectores en esta partición.	4 Bytes

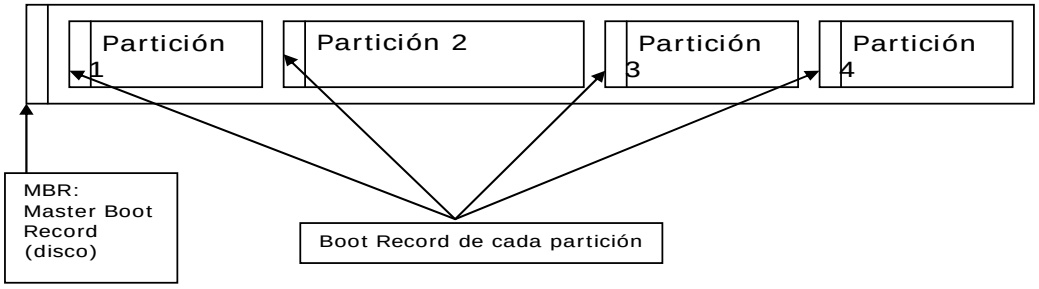
Contenido de cada una de las 4 entradas de la tabla de particiones.
Longitud = 10h = 16 Bytes.

Vemos el campo que se usa para indicar si esta partición es la activa o no, y también como se indican las fronteras de inicio y fin de la partición. Estas fronteras se indican usando el direccionamiento CSH de un disco duro (Cilindro, Sector, Cabeza). También se indican por cada partición el tipo de partición que es (esto nos permitirá que no existan problemas al instalar el sistema operativo correspondiente), la dirección del primer sector de la partición o sector de arranque para cederle el control de la CPU si es necesario, y un campo de comprobación donde se indican el numero de sectores totales de la partición.

Las particiones de un disco duro pueden ser de dos tipos:

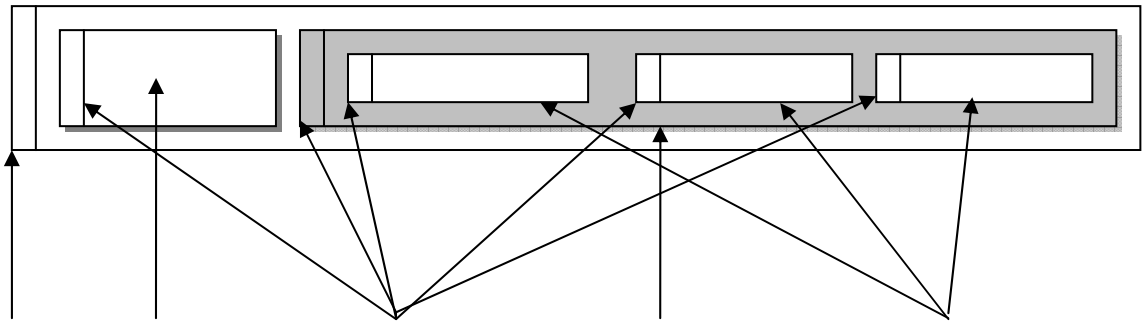
1. Primarias
2. Extendidas.

En un disco duro puede haber 4 particiones como máxima, lo que implica que puede haber 4 particiones primarias como máximo. Sin embargo, no puede haber más de 1 partición extendida en un disco duro (En realidad, si es posible tener más, pero mediante procedimientos especiales que no son compatibles con todos los sistemas operativos). Cada partición primaria forma un volumen (una letra de unidad, para entendernos) y tiene su propio sector de arranque. Una partición extendida sin embargo, no forma ningún volumen, ni tiene un sector de arranque como tal. Una partición extendida en realidad es un **contenedor de unidades lógicas**.



Cada unidad lógica que se crea dentro de una unidad extendida si forma su propio volumen, aunque no tiene un sector de arranque real, sino que usa su sector de arranque para controlar su tamaño entre otras cosas.

De esta manera, si dividimos un disco duro en una partición primaria (un volumen) y una partición extendida (donde creamos 10 unidades lógicas, cada una con su propio volumen) formaremos un total de 11 volúmenes (11 letras de unidad) pero solo tendremos un sector de arranque usable como tal, el de la partición primaria.



MBR Partición Primaria **Sectores arranque** Partición Extendida Unidades Lógicas

Solo el sector de arranque de una partición primaria es valido para arrancar el sistema operativo. El sector de arranque de la partición extendida solo contiene información sobre las unidades lógicas que se encuentran dentro de ella, y los sectores de arranque de las unidades lógicas contienen información específica a cada unidad lógica.

La tabla del MBR identifica la localización y tamaño de la partición extendida, pero no contiene información sobre las unidades lógicas creadas dentro de esta partición extendida. Ninguna de estas unidades lógicas pueden ser marcadas como activas, por lo que es posible que instalemos un sistema operativo en alguna de estas particiones lógicas, pero nunca podrá ser cargado directamente, ya que no podemos marcar esa partición como activa, y por lo tanto no podemos indicar que sea el disco de arranque. Si podemos cargar estos sistemas operativos instalados dentro de una unidad lógica, pero usando un gestor de arranque que haga las funciones del sector de arranque del que no disponen. Veamos como:

El truco esta en instalar un programa especial en el MBR. Este programa, conocido como gestor de arranque puede engañar a la maquina, buscando información sobre las particiones lógicas, y luego cargando el boot sector deseado en lugar del que debería leerse.

Estos programas, que permiten "hacer trampas" en el momento del arranque, suele ser conocidos como gestores de arranque (Boot Manager) y suelen venir incluidos junto con los sistemas operativos "profesionales", Windows XP/2000/2003, Linux, OS/2, Beos, UNIX, etc. Estos gestores permiten indicar en el momento del arranque, de cual volumen vamos a cargar el boot sector, sin importarles si dicho volumen es una partición primaria o una unidad lógica.

Windows 2000, 2003 y XP cuentan con su propio gestor de arranque **ntldr** que se instala automáticamente al instalar uno de estos sistemas operativos, pero solo se activa si detecta que en el disco duro existe mas de un sistema.

Por su parte, los sistemas basados en Linux utilizaban un gestor de arranque conocido como **LILO** (Linux Loader) aunque cada vez más sistemas Linux han cambiado este gestor por otro mucho más potente **GRUB** (Grand Unified Bootloader).

Todos estos gestores de arranque funcionan en modo texto normalmente. Nos presentan una lista con todos los sistemas operativos instalados en nuestros discos duro, y escogemos aquel con que deseemos cargar. Hay gestores que trabajan de forma gráfica, pero debido a su mayor tamaño no son especialmente recomendables.

La tabla de particiones, puede ser gestionada por diversos programas que se incluyen en los sistemas operativos. En sistemas como DOS y Windows 9x, la utilidad encargada de esto es el FDISK. En la familia Windows NT (NT, XP, 2000 y 2003) es el Administrador de discos (diskmgmt.msc). Linux por su parte incluye varios programas de este tipo, como pueden ser fdisk, qtparted, parted, etc. Hay que indicar que el FDISK de DOS y Windows 9x es una utilidad muy limitada, sin las características avanzadas que suelen tener este tipo de utilidades.

Windows permite indicar que letra de unidad se le asignará a cada partición, sin embargo DOS y Windows asignan estas letras por defecto. Primero, la C: es asignada a la partición primaria del primer disco donde se encuentre un sistema de ficheros FAT. Entonces la siguiente letra es asignada a la partición primaria con FAT del segundo disco, etc. Una vez acabadas con las particiones primarias de cada disco, se empiezan a asignar letras a las unidades lógicas del primer disco, luego a las unidades lógicas del segundo disco, etc. Una vez acabado con las unidades lógicas se continúa con el resto de particiones primarias que queden.

Veamos un ejemplo sobre esto. Un usuario tiene un único disco duro dividido en una partición primaria (C:) y un volumen lógico en una partición extendida (D:). Ahora este mismo usuario compra un segundo disco duro y lo instala, creando en el otra partición primaria y otra partición extendida, conteniendo otro volumen lógico. Pues bien, después de encender el ordenador, la partición primaria del segundo disco se llama (D:). El volumen lógico del primer disco, que antes se llamaba D pasa a llamarse (E:) y por fin, el volumen lógico del segundo disco recibe el nombre de (F:). Este tipo de cambios es muy peligroso, ya que al cambiar los nombres de las unidades es muy probable que muchos programas dejen de funcionar. Indicar que puesto que las unidades de CD reciben el nombre las ultimas, si este usuario instalase ahora un lector de CD, recibiría el nombre de (G:).

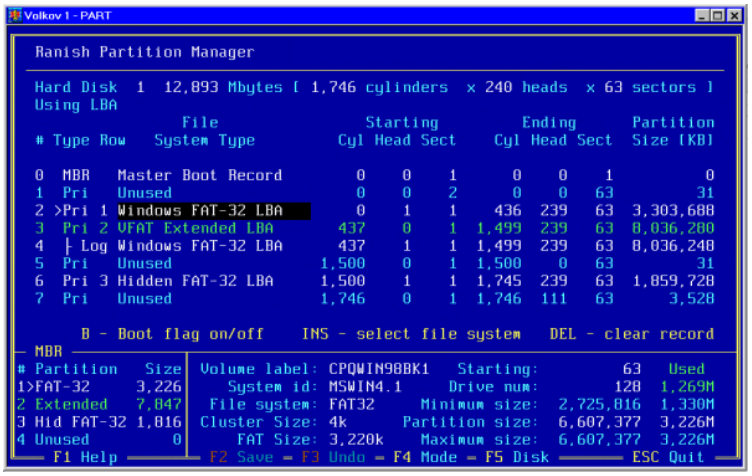
Este problema ocasionado por los sistemas operativos antiguos de Microsoft DOS y Windows 98 no esta presente en los sistemas operativos modernos de Microsoft. Así, por ejemplo, Windows XP asigna a cada unidad una letra según lo que hemos visto anteriormente, pero si se encuentra con una unidad que ya ha recibido nombre, no lo cambia.

Linux por su parte no presenta problemas de este tipo, ya que no asigna letras a los volúmenes, en su lugar tenemos que montar cada volumen en una directorio de nuestro árbol de directorios, por lo que no le afectan los problemas de nominación de volúmenes.

Hay que tener mucho cuidado al trabajar con las particiones. La tabla MBR es una tabla muy sensible a cualquier tipo de cambios. Una mala elección de cualquiera de sus campos, puede llevar a la inutilización total del disco duro. Además, dada la facilidad para "trastear" con la tabla de particiones, muchos programas utilizan configuraciones extrañas que son desconocidas para otros programas, lo que puede llevar a perder particiones o a cambiar su tamaño de modo incorrecto. Es altamente aconsejable no usar programas de gestión de tablas de particiones, excepto los que incluyen los propios sistemas operativos.

En caso de decidir usar un programa gestor de particiones, se recomienda usar un editor, mas que un gestor. Es decir, un programa que nos muestre directamente la tabla de particiones y nos permita retocarla como queramos, de forma manual y dejándolo todo bajo nuestro control. Uno de los mejores editores de particiones que podemos usar es el Ranish que además es gratuito. http://ranish.com. También es útil usar el diskedit de las antiguas utilidades norton que permiten acceder a bajo nivel a los discos, o una herramienta parecida. Sin embargo todas estas utilidades necesitan funcionar arrancando nuestro sistema desde un disquete, ya que si arrancamos Windows XP este automáticamente protege el disco duro para que nadie pueda acceder a el a bajo nivel.

Normalmente, no es nada aconsejable usar programas que editen las particiones, los sectores de arranque, el MBR, etc. de forma automatizada mediante asistentes. Programas del tipo Partition Magic toman muchas decisiones por nosotros, y si bien funcionan sin problemas en sistemas simples, en un sistema con una estructura complicada suelen cometer errores que normalmente conllevan problemas de gran magnitud.



2.3 ARRANQUE DE WINDOWS XP/2000/ 2003

1. Se carga y ejecuta el **POST**
2. Se carga el **MBR** del disco duro
3. Se carga el **sector de arranque** de la partición primaria activa
4. Se carga el programa **NTLDR**
5. **NTLDR** ajusta el procesador para trabajar a 32 bits
6. **NTLDR** lee el fichero **BOOT.INI** y muestra un menú si es necesario
7. El usuario selecciona un sistema operativo del menú, o se carga por defecto uno de ellos
8. **NTLDR** carga **NTDETECT.COM**
9. **NTDETEC.COM** genera la lista de hardware. Devuelve el control a **NTLDR**
10. **NTLDR** carga **NTOSKRNL.EXE**
11. **NTOSKRNL.EXE** lee el registro de Windows, y procede a ir cargando el sistema completo.

NTOSKRNL.EXE como indica es en gran parte el kernel o núcleo del sistema operativo, y es un programa de gran tamaño que se encuentra normalmente en nuestro directorio Windows. Sin embargo, tanto ntldr, como boot.ini o ntdetect.com son programas pequeños.

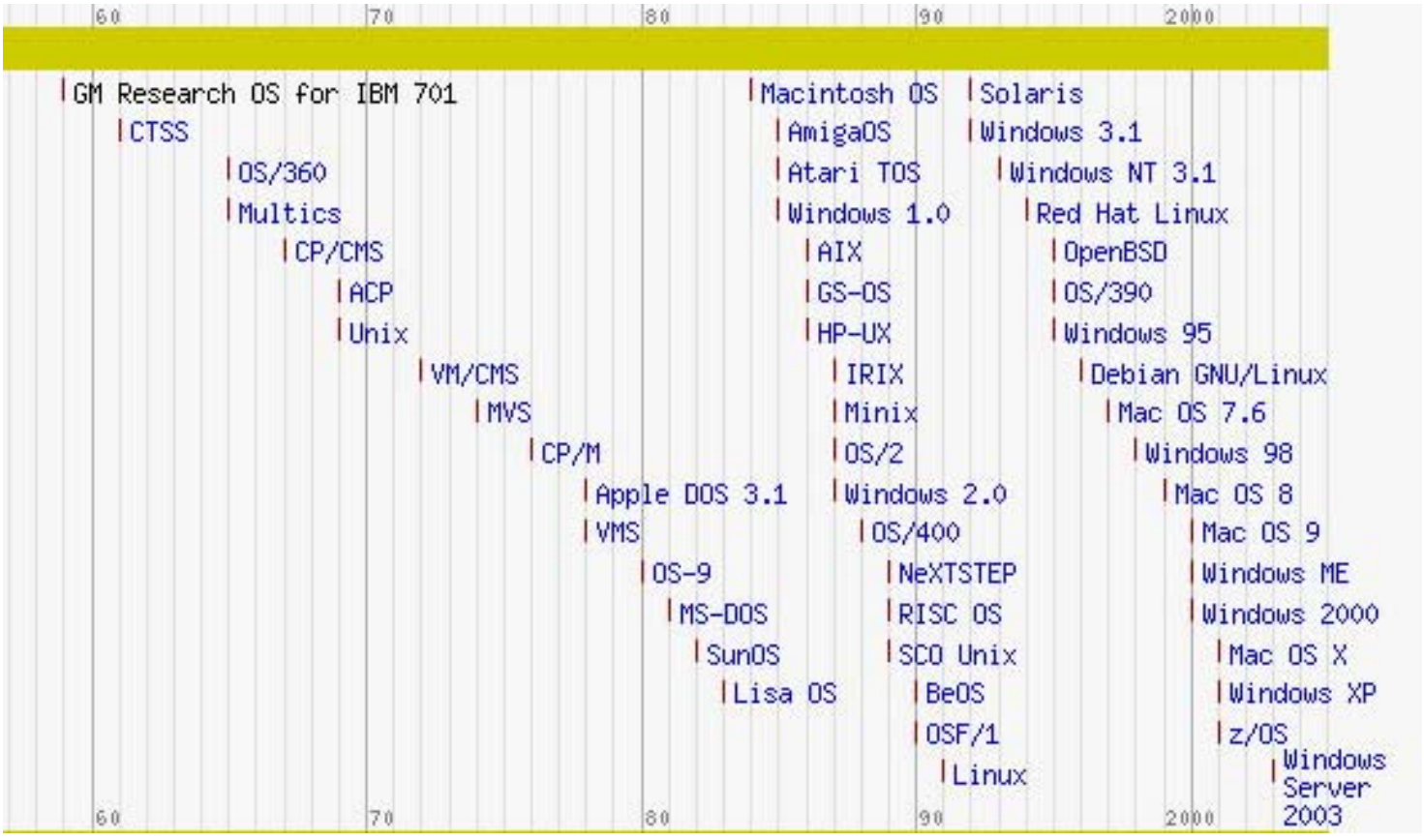
Esto permite que podemos situar dichos ficheros en un disquete, llavero usb, etc, con lo que tendríamos un volumen de INICIO, lo que nos permitiría iniciar el sistema aunque el disco duro haya sufrido algún problema. Sin embargo, no se puede confundir este "disco de inicio" con un "disco de arranque". Cuando llegue el momento de cargar NTOSKRNL.EXE si no se encuentra, el sistema se detendrá y no arrancará, y por el tamaño de dicho fichero y de todos los que necesita para trabajar, es imposible copiarlo en un volumen si no es de gran tamaño.

Es importante conocer esta secuencia, para comprender los distintos errores que se pueden cometer y con los que nos podemos encontrar. Por ejemplo, si recibimos el mensaje "falta ntldr" al intentar arrancar, esta claro que se ha producido un error en el punto 4, lo que nos indicaría que se ha leído el MBR, el sector de arranque, y no se ha encontrado en el raíz de nuestro volumen el fichero ntldr, bien por que lo hayan borrado o por que se haya borrado todo el volumen.

Sin embargo, un mensaje "falta ntoskrnl.exe" nos indicaría que si existe un fichero ntldr, pero que en nuestro directorio de Windows no se ha encontrado un fichero NTOSKRNL.EXE.

3 Historia de los sistemas operativos.

Vemos aquí una lista cronológica indicando en que momento aparece cada sistema operativo.



Los sistemas operativos pueden ser agrupados según su tecnología (sistemas tipo unix, sistemas tipo Windows, etc.), por su filosofía de propiedad (propietario o código libre), por su estado actual (histórico o actual), por su aplicación, etc.

Algunos sistemas operativos, agrupados según su desarrollador o propietario:

- **Apple/Macintosh**:
  - Apple DOS, ProDOS, GS/OS, Lisa OS, A/UX, Mac OS, Mac OS X, Mac OS X Server, Darwin
- **Microsoft**:
  - MS-DOS, Windows CE, Windows (1.0, 2.0, 3.0, 3.1, 95, 98 Me), OS/2 Windows NT (NT 3.5, NT 4.0, 2000, XP, 2003, Longhorn (Vista), Blackcomb) Windows Media Center, Xenix, Pocket PC, etc.
- **IBM**:
  - PC-DOS, OS/2, BOS, TOS, OS/360, DOS/360, DOS/VSE, VM/CMS, MFT, MVT, SVS, MVS, TPF, OS/390, z/OS, OS/400, AIX, ALCS, IBSYS, DPPX, K42
- **Sun Microsystems**:
  - Solaris, SunOS, Java Desktop System
- **Digital/Compaq/HP**:
  - AIS, OS-8, ITS, WAITS, TENEX, RSX-11, RT-11, VMS
- **Be Incorporated**:
  - Beos, Zeta
- **AT&T**:
  - System V
- **SCO/Caldera**:
  - SCO UNIX
- **Bell Labs**:
  - UNIX
- **Psion**:
  - EPOC, Symbian OS
- **Palm**:
  - Palm OS
- **Netware**:
  - Netware
- **Open Source** (Codigo abierto):
  - BSD, FreeBSD, DragonFly BSD, NetBSD, OpenBSD, Linux Red Hat, Suse, Debian, Knoppix, Guadalinex, Ubuntu, GNU Hurd, SSS-PC, etc.