**ENTORNO DE UN PROGRAMA**

Todos los sistemas operativos, y Windows 7 no es una excepción, tienen un área de datos llamada "entorno". No es nada más que un área donde se guardan ciertas variables con su contenido.

Es importante entender que cada programa de Windows tiene su entorno (igual o diferente a otro programa). Los entornos se heredan. Cada programa (y el propio intérprete de comandos, cmd.exe, es un programa más) cuando se lanza, "hereda" dicho entorno. Por heredar, no quiere decir que "use" el mismo que el programa padre, sino que al lanzarse, el "loader" -cargador- del sistema operativo realiza una copia del entorno padre en una nueva área de datos y al lanzar el programa le da como dirección del área de entorno esa nueva copia del entorno del "padre".

En otras palabras, cualquier modificación en las variables de entorno dentro de un programa no afecta al sistema ni al resto de programas, ya que lo que haría es modificar su propio entorno: la copia del entorno original del programa padre.

El sistema operativo al cargarse predefine ya una serie de variables. Podemos verlas, bien con botón derecho en **Equipo>propiedades>Configuración avanzada del sistema>pestaña Opciones azanzadas>botón de variables de entorno**, o bien de una manera más simple, lanzando el intérprete de comandos (cmd.exe) y tecleando el comando "set" (sin comillas).

Realmente, aunque lo veamos en conjunto, existen dos entornos: uno del sistema y uno de usuario, pero la visión de ambos es el conjunto de los dos.

Acabamos de ver nuestro primer comando: "set". Este comando nos permite no sólo ver todas las variables, sino también definir, cambiar, borrar su contenido y algunas opciones más.

Si lo ejecutamos por consola, nos dará algo similar a:

ALLUSERSPROFILE=C:\Documents and Settings\All Users  
APPDATA=C:\Documents and Settings\mi usuario\Datos de programa  
CommonProgramFiles=C:\Archivos de programa\Archivos comunes  
COMPUTERNAME=MIMÁQUINA  
ComSpec=C:\WINDOWS\system32\cmd.exe  
FP\_NO\_HOST\_CHECK=NO  
HOMEDRIVE=C:  
HOMEPATH=\Documents and Settings\mi usuario  
NUMBER\_OF\_PROCESSORS=1  
OS=Windows\_NT  
Path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;.....etc  
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH  
PROCESSOR\_ARCHITECTURE=x86  
PROCESSOR\_IDENTIFIER=x86 Family 6 Model 7 Stepping 3, GenuineIntel  
PROCESSOR\_LEVEL=6  
PROCESSOR\_REVISION=0703  
ProgramFiles=C:\Archivos de programa  
...etc...

Fijémonos que la estructura es: nombre de variable=contenido de la variable

Las variables, dentro de una consola de comandos o bien dentro de un script se referencian para poder ver su contenido encerradas entre símbolos de %. Por ejemplo, en el caso anterior, para ver el contenido de la variable COMPUTERNAME, simplemente podemos ejecutar:

**echo %COMPUTERNAME%**

Esto nos dará como resultado:

MIMÁQUINA

Igualmente podríamos cambiarlo con el comando set citado anteriormente:

**set COMPUTERNAME=nuevoNOMBRE**

Pero...

Esto realmente tiene el efecto de cambiar el nombre del ordenador? por supuesto que no. Esto sólo cambia el contenido de dicha variable. ¿Dónde lo cambia? pues tal y como hemos comentado anteriormente, lo cambia en el entorno del programa, es decir, en la copia del entorno original heredado por el programa. En nuestro caso, al ejecutarlo desde un cmd.exe, implica que cambia el entorno del cmd.exe (y sólo de él, es decir, si tuviésemos lanzadas dos consolas de comandos, cada una de ellas con cmd.exe, sólo se cambiaría en el cmd.exe que hayamos ejecutado ese comando "set"). Y ¿para que puede servir? simple, si recordamos que un programa hereda una copia del entorno del programa que lo lance, esto implicará que todo lo que lancemos desde esa consola de comandos, tendrá el contenido de esa variable modificado.

Cómo podemos crear una nueva variable de entorno? tan simple como darle un nombre y asignarle su contenido. Por ejemplo:

**set nuevo=prueba**

Esto creará si no existe, o modificará el contenido si existiese, de una variable de entorno llamada "nuevo", y le asignará el contenido de "prueba".

Podemos asignar a una variable el contenido de otra?: sí, por supuesto. Simplemente recordando que el contenido de una variable es precisamente el nombre de la variable encerrada entre símbolos %.

**set otra=%nuevo%**

Esto creará la variable "otra" con el contenido de la variable "nuevo", el cual era el texto "prueba". Si después de ejecutar el comando anterior realizamos:

**echo %otra%**

su resultado será:

prueba

Cómo podemos borrar -eliminar- una variable? simplemente no asignando nada.

**set otra=**

En este caso borrará la variable. Si ahora damos el comando "set" que nos muestra todas las variables, la variable "otra" ya no aparecerá.

Puedo concatenar textos con variables en una asignación?: Sí, por supuesto. Por ejemplo:

**set otra\_de\_nuevo=Esto es una %nuevo% de concatenación**

El resultado de lo anterior será que la variable "otra\_de\_nuevo" contendrá el valor "Esto es una prueba de concatenación".

Podemos tener contenidos numéricos y no sólo alfabéticos, y por tanto realizar operaciones matemáticas con el comando set? sí, se puede. La manera es con el modificador /a. Pongamos un ejemplo:

**set n=234** (asignar 234 a la variable "n")

**set /a i=%n%/4**

Veremos que esto nos devuelve por pantalla el valor 58 (es decir la división de 234/4) y además se lo asigna a la variable "i".  
Es necesario el modificador /a, ya que si no lo hubiésemos puesto, la expresión:

**set i=%n%/4**

no hubiese realizado la operación matemática y se hubiese quedado con el contenido (como literal) de:

234/4

es decir, una cadena de caracteres con ese contenido y no con el resultado matemático de la división.

Fijémonos, por curiosidad en este último caso y se puede ir realizando una prueba en una consola de comandos.

Realicemos:

**set n=234  
set i=%n%/4  
echo %i%**

veremos entonces:

234/4

Si ahora realizamos:

**set/a j=%i%**

La asignación a una variable numérica, puede ser decimal, hexadecimal u octal.  
Podemos hacer:

**set/a i=14  
set/a i=0x0E**

en ambos casos contendrá el decimal 14 (El valor hexadecimal 0E equivale al decimal 14).

O bien, si queremos introducir un numero en octal, simplemente lo empezamos por cero.

**set/a i=021**

El valor octal 021 es el decimal 17. Podéis comprobarlo en pantalla simplemente tecleando la línea anterior y viendo el resultado. L**os resultados los expresará siempre en decimal.**

Esto puede ser útil para convertir directamente de hexadecimal o bien de octal a decimal de una manera rápida.

Ya que hemos empezado con el comando "set" vamos a finalizar todo lo que dicho comando puede hacer. Además de lo anterior:

**set p**

mostrará en pantalla "todas" las variables de entorno que empiecen por p.   
Y para finalizar el modificador /p. Es decir, una sintaxis del estilo:

**set/p variable=[literal]**

Mostrará el "literal" en pantalla y el cursor se queda a continuación esperando que escribamos el valor del dato. Cuando lo escribimos y tecleamos "intro", lo que hayamos tecleado se asignará a la variable de entorno definida en el set. Por ejemplo:

**set /p dato=Introduce datos:**

Nos mostrará por pantalla:

Introduce datos:\_

y cuando introduzcamos los datos los asignará a la variable "dato".

Esto es válido realizarlo en un script. En ese momento se detendrá la ejecución del script y nos pedirá datos por pantalla. Al pulsar la tecla "intro", se los asignará a dicha variable y continuará la ejecución del script.

**CARACTERES ESPECIALES**

Hay ciertos caracteres "especiales" que hay que usar con extremada precaución: por ejemplo, evitar usar, a no ser que lo necesitemos explícitamente, caracteres reservados como **&**, **>**, **<**, **|**, **%**, **=**, **^**.

Pongamos un ejemplo. El carácter & ("and" en inglés, es decir "y" en castellano) se interpreta como un "y", por tanto, ejecuta lo que hay antes del & "y" además a continuación, ejecuta lo que hay después del &. Es decir, si ejecutásemos:

**set var=a&a**

nos dará un error que veremos por pantalla:

***"a" no se reconoce como un comando interno o externo, programa o archivo por lotes ejecutable***

Qué ha hecho el sistema? Ha ejecutado el comando que hay antes del &, es decir:

**set var=a**

Por tanto le ha asignado a la variable de entorno "var" el contenido "a". A continuación, el intérprete de comandos intenta ejecutar lo que hay después del &, es decir intenta interpretar el comando "a". Como "a" no es un comando reconocido, ni un .exe, ni un .com, ni el nombre de otro script nos dará el error mostrado anteriormente. Curiosamente:

**set var=a&cmd**

¿qué hace? Ejecutadlo y ved el resultado.

Pero... ¿y si realmente queremos asignar el contenido del literal "a&a" a la variable "var", cómo lo hacemos? Simplemente anteponiendo el carácter ^ a cualquiera de los caracteres especiales. En nuestro caso deberíamos haber hecho:

**set var=a^&a**

Si posteriormente ejecutamos un:

**echo %var%**

nos seguirá mostrando un error, ya que aunque realmente tiene el contenido "a&a", nos mostrará "a" e intentará ejecutar lo que va a continuación: intentará la ejecución de un comando llamado "a".   
Para ver que realmente tiene el contenido solicitado, podemos hacer dos cosas:

**set v**

nos mostrará el contenido de todas las variables que empiezan por "v" y podremos comprobar que la variable "var" contiene lo que realmente deseábamos. O bien, redirigir la salida a un archivo (los símbolos de redirección y su significado los veremos más adelante).

Un error bastante corriente cuando se usan caracteres de asignación y concatenación (&) en la misma línea puede ser:

**set var=prueba & set var1=otra\_prueba**

¿Cuál es, o puede ser, el error cometido? pues...recordemos que todo lo anterior al & se ejecuta como un comando. Es decir: ¿qué hemos asignado a la variable "var"? podemos suponer que "prueba". Pero no es así, hemos asignado todo lo anterior al &, es decir "prueba ", con un espacio en blanco al final. Esto nos puede causar problemas posteriores si lo que queríamos hacer era asignar sólo "prueba".

Podemos verificarlo tecleando entre algún delimitador, por ejemplo un \*:

echo \*%var%\*

mostrará un \*, el contenido de var y otro \*. Obtendríamos:

\*prueba \*

con el fatídico (en este caso) carácter en blanco al final.

**OPERACIONES CON VARIABLES DE ENTORNO**

Hemos visto hasta el momento que las variables de entorno no son nada más que cadenas de caracteres, también es verdad que mediante algún modificador especial en el comando set (**/a**) pueden tratarse como numéricas para realizar algunas pequeñas operaciones matemáticas.

Vamos a ver primero las funciones que podemos aplicar a las variables de caracteres y en segundo lugar las operaciones matemáticas que podemos hacer así como su precisión cuando el contenido es numérico. Igualmente veremos las llamadas variables dinámicas de entorno: variables que aunque no veamos mediante el comando "set" pueden usarse en instrucciones de comandos.

Tengamos siempre presente que los caracteres "especiales" deben llevar el símbolo ^ precediéndolos. Por ejemplo, un error que viene en la propia ayuda de Windows del comando set, es que la propia ayuda nos dice: "El comando SET no permitirá que un signo de igual sea parte de una variable". Esto no es "del todo" verdad. Si hacemos:

**set var=a^=b**

la variable "var" quedará con el contenido "a=b". Se puede comprobar si a continuación de lo anterior ejecutamos:

**set v**

**1-OPERACIONES SOBRE VARIABLES DE CARACTERES**

Extraer una subcadena

**%var:~n,m%**

Esto nos extrae de la variable "var", la subcadena desde la posición "n" con longitud "m". "n" es el desplazamiento empezando a contar desde 0, por tanto, la primera posición es la 0, la segunda la 1, etc.

Pongamos un ejemplo:

**set var=123456789**

**echo %var:~1,4%**

nos mostrará la subcadena de "var" que va desde la posición 2 (recordemos: con desplazamiento +1 ya que empieza por 0) y con longitud 4. Es decir, veremos: "2345".

Tanto n como m, son opcionales. Es decir, por ejemplo:

**%var:~1%**

nos mostrará desde la posición 2 (offset de 1), hasta el \*final\* de la variable (ya que no se ha especificado longitud). Por tanto, mostrará en el ejemplo anterior: "23456789".   
Igualmente:

**%var:~,5%**

al no especificarse posición se asume desde el principio de la cadena y por tanto nos mostrará 5 caracteres iniciales. En nuestro caso "12345".

si "n" es un número negativo, se refiere a la longitud de toda la cadena menos ese número. Por ejemplo:

set var=123456789

**echo %var:~-2%**

nos mostrará los dos últimos caracteres de la cadena, es decir "89".

Igualmente, si "m" es negativo, se refiere a \*toda\* la cadena menos "m" caracteres.

set var=123456789

**echo %var:~,-2%**

nos mostrará todo menos los dos últimos caracteres de la cadena, es decir "1234567".

Sustituir dentro de una cadena, un literal por otro

**%var:str1=str2%**

Este comando buscará todas las ocurrencias de la subcadena "str1" dentro de "var" cambiándolas por "str2".  
Por ejemplo:

set var=hola cómo estás

Ejecutad para verlo:

**echo %var:cómo=que tal%**

Igualmente podríamos asignárselo de nuevo a "var" o bien a otra nueva variable de entorno. Por ejemplo:

**set var=%var:cómo=que tal%**

con lo cual el contenido habría quedado modificado.

Puede omitirse "str2". En ese caso la cadena str1 dentro de la variable var quedará eliminado. Por ejemplo:

**set var=12345xx6**

**set var=%var:xx=%**

**echo %var%**

Se cambiarán **todas** las ocurrencias de "str1". Si "str1" figura varias veces se cambiará todas las veces en la cadena de caracteres.

**FUNCIONES MATEMÁTICAS SOBRE VARIABLES DE ENTORNO**

Con el modificador /a en el comando set podemos realizar operaciones matemáticas \*simples\* sobre variables.

Debemos primero hablar de la precisión: sólo se opera con números enteros. Por tanto la división de 5/2 nos dará como contenido un 2. Igualmente la máxima precisión es 2^32.

En general la sintaxis es:

set /a var=expresión matemática.

Por "expresión matemática" puede entenderse cualquiera de las siguientes y en orden de precedencia de mayor a menor (extraído de la propia ayuda del comando set):

**()** - agrupar  
**! ~ -** - operadores unarios  
**\* / %** - operadores aritméticos  
**+ -** - operadores aritméticos  
**<< >>** - desplazamiento lógico (bit a bit)  
**&** - bit a bit y  
**^** - bit a bit exclusivo o  
**|** - bit a bit  
**= \*= /= %= += -=** - asignación  
**&= ^= |= <<= >>=**  
**,** - separador de expresión

Si usamos una variable de entorno dentro de una expresión matemática y dicha variable no existe, será asumido que contiene un cero.

Veamos los matices de lo anterior, ya que si no tenemos en cuenta algunas matizaciones ya comentadas en este documento habrá expresiones que nos darán error.

Volvamos a recordar que hay símbolos especiales que no pueden escribirse directamente, por ejemplo: &, <, >, |, o incluso ^. Cada uno de ellos debe ir precedido por un ^. Es decir, el bit a bit exclusivo ^, debe escribirse como ^^ ya que si no el intérprete de comandos lo interpretará erróneamente.

Veamos ejemplos de "asignaciones".

set /a x=x+1

podemos sustituirlo (abreviarlo) por una notación similar al C o C++, es decir:

set /a x+=1

(a la variable "x" sumarle 1 y asignárselo a "x")

Lo mismo:

set /a x=x-1 es equivalente a: set /a x-=1  
set /a x=x\*23 es equivalente a: set /a x\*=23

etc... para el resto de operaciones matemáticas de asignación citadas anteriormente.

Veamos el siguiente problema: ¿qué solución tiene?

set /a x=8, y=2, z=16

set /a z-=(x-y)/2+5^<^<2

Quizá no sea tan trivial el ver que el resultado de lo anterior es -16. Intentad razonarlo.

Aunque no lo he encontrado documentado, pueden hacerse operaciones del tipo:

set /a x=(y=3+4)\*(z=2+1)  
x vale 21, y vale 7, z vale 3

**VARIABLES DINÁMICAS DE ENTORNO**

Son variables que aunque no veamos mediante el comando "set" tienen contenido que va variando o puede variar dinámicamente.   
Un ejemplo de esto es la variable %TIME%. Cada vez que ejecutemos:

echo %TIME%

nos dará dinámicamente la hora del sistema.

Estas variables son (autoexplicativas desde la ayuda del comando set):

**%CD%** - se expande a la cadena del directorio actual.

**%DATE**% - se expande a la fecha actual usando el mismo formato que el comando DATE.

**%TIME**% - se expande a la hora actual usando el mismo formato que el comando TIME.

**%RANDOM%** - se expande a un número decimal aleatorio entre 0 y 32767.

**%ERRORLEVEL%** - se expande al valor de NIVEL DE ERROR actual

**%CMDEXTVERSION%** - se expande al número actual de versión de las extensiones del comando del procesador (lo veremos posteriormente)

**%CMDCMDLINE%** - se expande a la línea de comando original que invocó el Procesador de comandos.