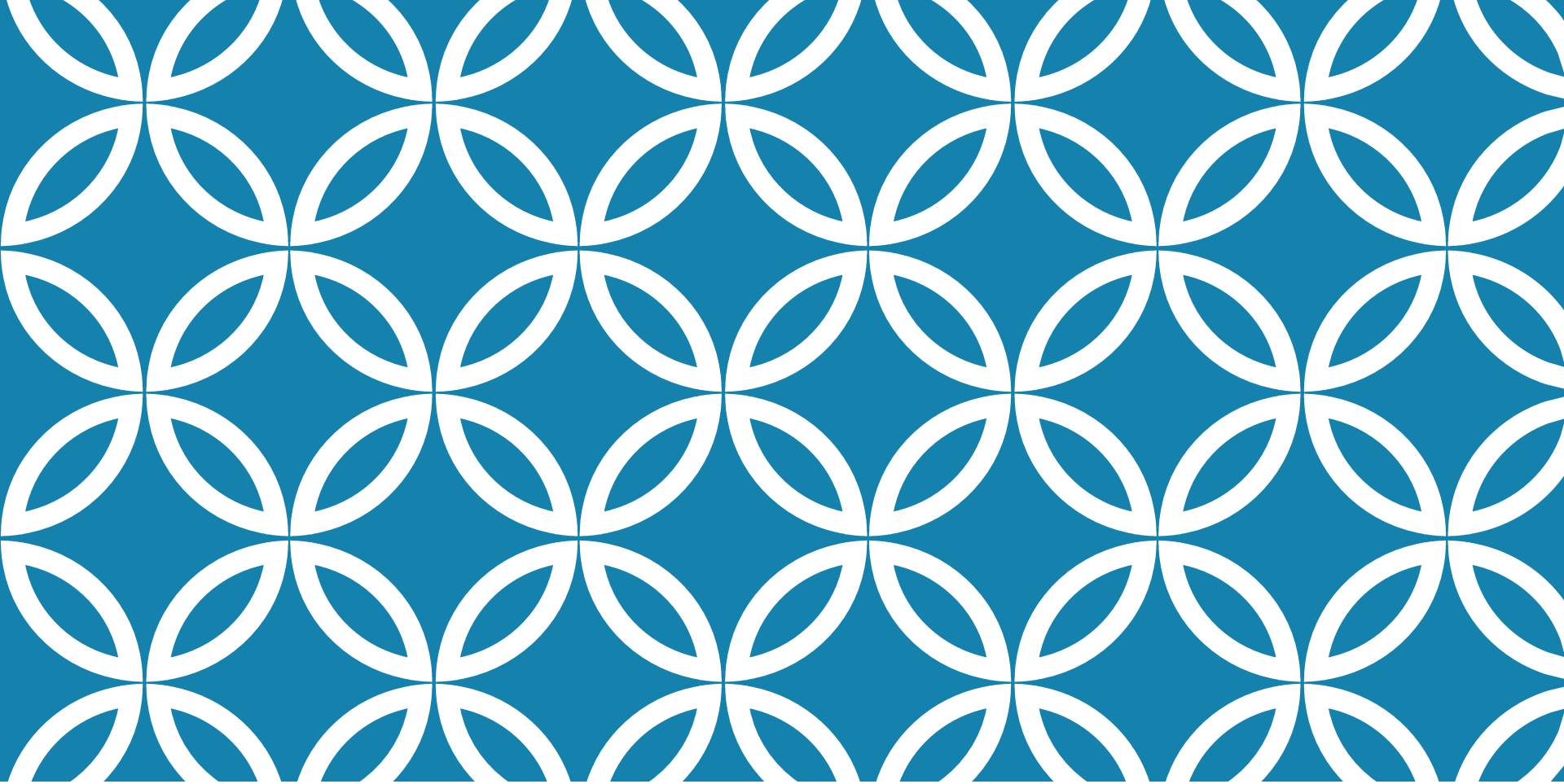

LENGUAJE DE COMANDOS - SCRIPTS .CMD .BAT

Windows

¿QUÉ ES UN SCRIPT EN LENGUAJE DE COMANDOS?

No es nada más que un fichero de texto, que puede generarse con el simple cuaderno de notas, y cuya extensión es .bat o .cmd.

ENTORNO DE UN PROGRAMA

Todos los sistemas operativos, y Windows 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.

Los entornos se heredan. Cada programa (y el propio intérprete de comandos, cmd.exe, es un programa más) cuando se lanza, "hereda" dicho entorno.

ENTORNO DE UN PROGRAMA

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".

ENTORNO DE UN PROGRAMA

El sistema operativo al cargarse predefine ya una serie de variables de entorno.

Podemos verlas, bien con botón derecho en Mi PC / propiedades / pestaña de opciones avanzadas / botón de variables de entorno.

ENTORNO DE UN PROGRAMA

Podemos verlas, bien con botón derecho en Equipo / propiedades / configuración avanzada de sistema/ pestaña de opciones avanzadas / botón de variables de entorno.

O bien de una manera más simple, lanzando el intérprete de comandos (cmd.exe) y tecleando el comando **set**.

ENTORNO DE UN PROGRAMA

Si lo ejecutamos por consola el comando set, 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

ENTORNO DE UN PROGRAMA

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...

La estructura es: nombre de variable=contenido de la variable

ENTORNO DE UN PROGRAMA

Las variables, dentro de una consola de comandos o bien dentro de un script se referencia para poder ver su contenido encerradas entre símbolos de %.

- echo %COMPUTERNAME%

Igualmente podríamos cambiarlo con el comando set citado anteriormente:

- set COMPUTERNAME=nuevoNOMBRE

ENTORNO DE UN PROGRAMA

¿Podemos asignar a una variable el contenido de otra?

Sí, simplemente recordando que el contenido de una variable es precisamente el nombre de la variable encerrada entre símbolos %.

- set otra=%nuevo%

ENTORNO DE UN PROGRAMA

¿Cómo podemos borrar -eliminar- una variable?

Simplemente no asignando nada.

- set otra=

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

- set otra_de_nuevo=Esto es una %nuevo% de concatenación

ENTORNO DE UN PROGRAMA

¿Podemos tener contenidos numéricos y no sólo alfabéticos, y por tanto realizar operaciones matemáticas con el comando set?

Sí, 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 lo asigna a la variable "i".

ENTORNO DE UN PROGRAMA

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 (recordemos que el hexadecimal 0E equivale al decimal 14).

ENTORNO DE UN PROGRAMA

- O bien, si queremos introducir un numero en octal, simplemente lo empezamos por cero.
  - > set /a i=021
- El octal 021 es el decimal 17. Realiza:
  - > Echo %i%
- **set carcter**
- Nos mostrará en pantalla "todas" las variables de entorno que empiecen por ese carácter.
  - > Set p

ENTORNO DE UN PROGRAMA

- El modificador /p del comando set. Es decir, una sintaxis del estilo:
- set /p variable=[literal]
- Lo que hace es muestra el "literal" en pantalla y el cursor se queda a continuación esperando que metamos un dato. Cuando lo metemos 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:_

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 **&, >, <, |, %, =, ^.**
- ¿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.

CARACTERES ESPECIALES

- En nuestro caso deberíamos haber hecho:

  ```
  >set var=a^&a
  ```

- Si posteriormente ejecutamos un:

  ```
  >echo %var%
  ```

- nos 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:

  ```
  >set v
  ```

OPERACIONES CON VARIABLES DE ENTORNO

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.

OPERACIONES CON VARIABLES DE ENTORNO

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. Igualmente:

- %var:~,5%

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

OPERACIONES CON VARIABLES DE ENTORNO

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".

OPERACIONES CON VARIABLES DE ENTORNO

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".

OPERACIONES CON VARIABLES DE ENTORNO

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

OPERACIONES CON VARIABLES DE ENTORNO

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%

OPERACIONES CON VARIABLES DE ENTORNO

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%

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

OPERACIONES CON VARIABLES DE ENTORNO

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:

()	- agrupar
! ~ -	- operadores unarios
* / %	- operadores aritméticos
+ -	- operadores aritméticos

OPERACIONES CON VARIABLES DE ENTORNO

Por "expresión matemática" puede entenderse cualquiera de las siguientes y en orden de precedencia de mayor a menor:

<< >> - desplazamiento lógico (bit a bit)

& - bit a bit y

^ - bit a bit exclusivo o

| - bit a bit

= *= /= %= += -= - asignación

&= ^= |= <<= >>=

, - separador de expresión

OPERACIONES CON VARIABLES DE ENTORNO

- 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

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%.

VARIABLES DINÁMICAS DE ENTORNO

Estas variables son:

%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

VARIABLES DINÁMICAS DE ENTORNO

Estas variables son:

%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

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

OPERADORES DE REDIRECCIÓN

- **>** Escribe la salida del comando (normalmente STDOUT) en un fichero o un dispositivo, en lugar de en la ventana del Símbolo del sistema.
  < Lee la entrada (STDIN) del comando desde un archivo, en lugar de leerla desde la consola.
- **>>** Añade la salida del comando al final de un archivo sin eliminar la información que ya está en él.
- **>&** Escribe la salida de un controlador en la entrada de otro controlador
- **<&** Lee la entrada desde un controlador y la escribe en la salida de otro controlador.
- **|** Lee la salida de un comando y la escribe en la entrada de otro comando.

OPERADORES CONDICIONALES

Son los operadores **&&** y el operador **||**

Lo que exista después del operador && se ejecutará "sólo" si la instrucción previa a él -en la misma línea- ha terminado correctamente (código 0).

Por contra, si usamos || sólo se ejecutará si la terminación de la instrucción previa termina incorrectamente (código distinto de cero)

Por ejemplo:

- md kkk && echo "finalizado correcta la creación"

Sólo veremos el mensaje de aviso si la creación de la carpeta kkk ha sido posible.

NOMBRES LÓGICOS DE DISPOSITIVOS

Hay ciertos dispositivos que podemos usar en cualquier comando o dentro de un script con nombres reservados que se refieren a dispositivos:

CON se refiere a la consola.

LPT1 se refiere a la impresora.

NUL dispositivo nulo. Lo que se envíe a él, se pierde.

CONTROL DE FLUJOS DE EJECUCIÓN

Instrucción **GOTO** (ir a)

Permite en un punto determinado de nuestro script "saltar" a otro punto del script.

GOTO nombre_etiqueta

Instrucción **ECHO**, su objetivo es mostrar por el STDOUT el literal que exista a continuación.

CONTROL DE FLUJOS DE EJECUCIÓN

ECHO literal a escribir
ECHO. (un "." pegado a la instrucción sin separación: escribe una línea en blanco)
ECHO ON ECHO OFF

Si queremos suprimir la salida de las líneas y dejar solamente la salida de los comandos, lo primero que debemos decirle al script es que no queremos "ECO" de lo que va traduciendo.

@ECHO OFF

El símbolo @ como primer carácter en una línea del script indica que no queremos que esa línea se "vea" en la ejecución

CONTROL DE FLUJOS DE EJECUCIÓN

Instrucción **IF** (si). Recordemos que lo encerrado entre corchetes es opcional

if [not] errorlevel número comando [else expresión]

if [not] cadena1==cadena2 comando [else expresión]

if [/i] cadena1 operadorDeComparación cadena2 comando [else expresión]

El /i realiza la comparación de tal manera que no se distingue entre mayúsculas y minúsculas.

if [not] exist nombreDeArchivo comando [else expresión]

OPERADORES DE COMPARACIÓN

Especifica un operador de comparación de tres letras. En la siguiente tabla puede ver los valores de operadorDeComparación.

EQU igual a (es lo mismo que ==)
NEQ no es igual a
LSS menor que
LEQ menor que o igual a
GTR mayor que
GEQ mayor que o igual a

CONTROL DE FLUJOS DE EJECUCIÓN

COMANDO 'FOR'

Es el comando más importante y más potente que podemos usar en un script. Ejecuta un comando especificado para cada archivo de un conjunto de archivos.

CONTROL DE FLUJOS DE EJECUCIÓN

for %variable in (grupo) do comando

%variable
Requerido. Representa un parámetro reemplazable. Utilice %variable para ejecutar for en el símbolo del sistema. Utilice %%variable para ejecutar el comando for dentro de un archivo por lotes. Las variables distinguen entre mayúsculas y minúsculas y se deben representar con un valor alfabético, como %A, %B o %C.

CONTROL DE FLUJOS DE EJECUCIÓN

for %variable in (grupo) do comando

(grupo)
Requerido. Especifica uno o varios archivos, directorios, intervalo de valores o cadenas de texto que se desea procesar con el comando especificado. Los paréntesis son obligatorios. Puede utilizar caracteres comodín (es decir, * y ?) para especificar un grupo de archivos.

(*.doc)
(*.doc *.txt *.me)
(ene*.doc ene*.rpt feb*.doc feb*.rpt)
(ar??1991.* ap??1991.*)

CONTROL DE FLUJOS DE EJECUCIÓN

for /D %variable in (grupo) do comando

/D -> Sólo directorios

Si grupo contiene caracteres comodín (* y ?), el comando especificado se ejecuta para cada directorio (en lugar de un grupo de archivos de un directorio especificado) que coincida con grupo.

CONTROL DE FLUJOS DE EJECUCIÓN

for /R [unidad :]rutaDeAcceso] %variable in (grupo) do comando [opcionesDeLíneaDeComandos]

/R -> Recursividad. Recorre el árbol de directorios con raíz en [unidad:]rutaDeAcceso y ejecuta la instrucción for en cada directorio del árbol. Si no se especifica un directorio después de /R, se considera el directorio actual. Si grupo tiene únicamente un punto (.) sólo se enumerará el árbol de directorios.

CONTROL DE FLUJOS DE EJECUCIÓN

for /L %variable in (númeroInicial,númeroPaso,númeroFinal) do comando

/L -> Iteración de un intervalo de valores. Se utiliza una variable iterativa para establecer el valor inicial (númeroInicial) y, después, recorrer un intervalo de valores especificado hasta que el valor sobrepase el valor final (númeroFinal) especificado. /L ejecutará la iteración mediante la comparación de númeroInicial con númeroFinal. Si númeroInicial es menor que númeroFinal, el comando se ejecutará. Si la variable iterativa sobrepasa el valor de númeroFinal, el shell de comandos sale del bucle.

CONTROL DE FLUJOS DE EJECUCIÓN

También se puede utilizar un valor númeroPaso negativo para recorrer un intervalo de valores decrecientes. Por ejemplo, (1,1,5) genera la secuencia 1 2 3 4 5 y (5,-1,1) genera la secuencia (5 4 3 2 1)

```
for /l %i in (5,-1,1) do @echo %i
```

CONTROL DE FLUJOS DE EJECUCIÓN

COMANDO 'FOR' INTERACCIÓN Y ANÁLISIS DE ARCHIVOS

El análisis de archivos se compone de la lectura de la información de salida, la cadena o el contenido del archivo, su división en líneas individuales de texto y el análisis de cada línea en cero o más testigos. Después, se llama al bucle for con la variable de iteración establecida al valor del testigo. De forma predeterminada, /F pasa el primer testigo separado por espacios en blanco de cada línea de cada archivo. Las líneas en blanco se omiten.

CONTROL DE FLUJOS DE EJECUCIÓN

Sintaxis:

for /F ["palabrasClaveDeAnálisis"] {%% | %}variable in (grupoNombreArchivos) do comando [opcionesDeLíneaDeComandos]

for /F ["palabrasClaveDeAnálisis"] {%% | %}variable in ("cadenaLiteral") do comando [opcionesDeLíneaDeComandos]

for /F ["palabrasClaveDeAnálisis"] {%% | %}variable in ('comando') do comando [opcionesDeLíneaDeComandos]

CONTROL DE FLUJOS DE EJECUCIÓN

El argumento grupoNombreArchivos especifica uno o varios nombres de archivo. Cada archivo se abre, lee y procesa antes de pasar al siguiente archivo en grupoNombreArchivos. Para suplantar el comportamiento predeterminado del análisis, especifique "palabrasClaveDeAnálisis". Se trata de una cadena incluida entre comillas que contiene una o varias palabras clave que especifican diferentes opciones de análisis.

CONTROL DE FLUJOS DE EJECUCIÓN

En la tabla siguiente se enumeran las palabras clave de análisis que se pueden utilizar para palabrasClaveDeAnálisis.

Palabra clave	Descripción
eol=c	Especifica un carácter de fin de línea (sólo un carácter).
skip=n	Especifica el número de líneas que se omitirán al principio del archivo.
delims=xxx	Especifica un grupo de delimitadores. Reemplaza el grupo de delimitadores predeterminado, formado por los caracteres espacio y tabulador.
tokens=x,y,m-n	Especifica los testigos de cada línea que se pasarán al cuerpo de for en cada iteración. Como consecuencia, se asignarán nombres de variable adicionales. La forma m-n es un intervalo que especifica los testigos m hasta n. Si el último carácter de la cadena tokens= es un asterisco (*), se asigna una variable adicional que contendrá el texto que queda en la línea después del último testigo analizado.
Usebackq	Especifica que se pueden utilizar comillas para incluir los nombres de los archivos de grupoNombreArchivos, que una cadena incluida entre comillas se ejecutará como si fuera un comando y que una cadena escrita entre comillas simples es un comando de cadena literal.

CONTROL DE FLUJOS DE EJECUCIÓN

Para ignorar n líneas de un archivo sería:

for /f "skip=n" %i in (pr.abc)

for %i in (*.abc *.txt) do @for /f "skip=1" %j in (%i) do @echo %j

Por tanto la instrucción:

for /f "skip=2 tokens=1,2,* delims= " %%g in ('tasklist /S %1 /U %user% /P %pass%') do...

Está recuperando en %%g el nombre de cada proceso y en %%h el PID de dicho proceso.

PARÁMETROS

Cualquier script es capaz de recibir parámetros desde la línea invocante.

Se considera parámetro cualquier literal enviado en la línea que invoca al script.

El sistema considera que los parámetros están separados por espacios en blanco o bien por el delimitador ";".

Cada parámetro se referencia dentro del script por %1, %2, %3, etc...

PARÁMETROS

Realmente podemos referenciar los parámetros desde %0 a %9.

El parámetro recibido en %0 no se teclea y siempre el sistema nos pasa como contenido el nombre del script invocado.

%1 a %9 son los posibles parámetros enviados en la línea de comandos.

El comando shift, el cual tiene por objeto desplazar los parámetros. Es decir, al ejecutar shift lo que se hace es que el %2 pasa a ser %1.

PARÁMETROS

A los parámetros y sólo a los parámetros, pueden anteponérseles modificadores los cuales realizan una conversión especifica. Realmente todos los modificadores posibles de parámetros son los que se describen a continuación:

%~1 Expande %1 y quita todas las comillas ("").
%~f1 Expande %1 y lo convierte en un nombre de ruta de acceso completo.
%~d1 Expande %1 a una letra de unidad.
%~p1 Expande %1 a una ruta de acceso.

PARÁMETROS

%~n1 Expande %1 a un nombre de archivo.

%~x1 Expande %1 a una extensión de archivo.

%~s1 Ruta de acceso expandida que únicamente contiene nombres cortos.

%~a1 Expande %1 a atributos de archivo.

%~t1 Expande %1 a una fecha/hora de archivo.

PARÁMETROS

%~z1 Expande %1 a un tamaño de archivo.

%~$PATH:1 Busca los directorios enumerados en la variable de entorno PATH y expande %1 al nombre completo del primer directorio encontrado. Si el nombre de la variable de entorno no está definido o la búsqueda no encuentra el archivo, este modificador se expande a la cadena vacía.

INVOCACIÓN A PROCEDIMIENTOS

Instrucción call

Muchas veces es necesario desde un script llamar a otro procedimiento, o llamar a una subrutina del mismo procedimiento.

* Llamada a procedimiento externo. Vamos a crearnos dos script. El primero de ellos lo llamaremos principal.cmd y con el contenido:

```
@echo off
echo estoy en el principal
call invocado.cmd
echo de vuelta en el principal
```

Y un segundo procedimiento llamado invocado.cmd:

```
@echo off
echo estoy en el invocado
```

INVOCACIÓN A PROCEDIMIENTOS

Llamada a procedimiento interno (subrutina). La sintaxis es “CALL :nombre”, en donde :nombre es un nombre de etiqueta interno al cual saltará la ejecución del script, pero a diferencia del goto, cuando finalice la propia rutina continuará la ejecución

Con goto :EOF. Esto indica que cuando se la invoca con CALL volverá a la línea de continuación del invocante.

INVOCACIÓN A PROCEDIMIENTOS

Pongamos un ejemplo (script.cmd)

```
@echo off
 echo Parámetro 1: %1
 echo Parámetro 2: %2
 call :rutina1 %2
 echo estoy de nuevo en el principal
 call :rutina1 %1
 echo estoy de nuevo en el principal
 call :rutina2 parametrosjuntos%1%2
 goto final

:rutina1
 echo en la rutina recibo parámetro: %1
 goto:EOF

:final
 echo estoy ya al final
```