**PARÁMETROS**

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

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 **;**(Punto y coma).   
Cada parámetro se referencia dentro del script por %1, %2, %3, etc...

Vamos a crear un script que reciba dos parámetros, el nombre de un fichero y el nombre de una carpeta. El objetivo del script es copiar el archivo desde la carpeta en donde estemos posicionados en la carpeta destino, siempre y cuando no exista ya en el destino.

Una manera grosera de realizar el script, llamémosle copia.cmd, sin verificar que los parámetros son correctos, sería:

**@echo off  
if not exist %2\%1 copy %2 %1**

Si invocásemos al script de la siguiente manera:

copia.cmd mifichero.txt c:\backup

al desarrollar la instrucción el script, realizaría:

if not exist c:\backup\mifichero.txt copy mifichero.txt c:\backup

que es precisamente lo que queríamos hacer. He comentado que este script, tal y como está, está realizado de una manera grosera. Si el nombre del archivo o el nombre de la carpeta destino tuviese espacios en blanco el sistema consideraría que dicho espacio en blanco es un delimitador de parámetros y por tanto nos llegarían al script más parámetros de los deseados. Realmente para realizar correctamente el script deberíamos hacer:

**@echo off  
if “%1”==““ goto error  
if “%2”==““ goto error  
if “%3” NEQ ““ goto error**

**if not exist %1 goto error1  
if not exist %2\nul goto error2  
if exist %2\%1 goto aviso  
copy %1 %2\%1 2>&1 >nul  
if not errorlevel 1 goto correcto**

**echo Copia errónea. Verifique permisos de la carpeta destino.  
goto fin**

**:error  
echo parámetros erróneos. Deben recibirse dos parámetros.  
goto fin**

**:error1  
echo Nombre del fichero origen no existe.  
goto fin**

**:error2  
echo Carpeta destino no existe.  
goto fin**

**:aviso:  
echo El archivo ya existe en el destino. No se ha realizado copia.  
goto fin**

**:correcto  
echo Archivo copiado correctamente.  
goto fin**

**:fin**

Y la manera de invocar al comando sería:

copia.cmd "nombre fichero.xxx" "c:\carpeta destino"

Hemos entrecomillado cada uno de los parámetros. De esta manera sólo se recibirán dos parámetros ya que si no, se recibirían 4 al usar como delimitador el sistema los espacios en blanco entre ellos.

Comentemos un poco el script.

\* Verificamos que hemos recibido 2 parámetros y sólo 2.  
\* Verificamos que exista el fichero origen  
\* Verificamos que exista la carpeta destino. Para verificarlo usamos un pequeño truco: preguntar por el dispositivo "nul" en la carpeta destino. Si no existe, es que la carpeta no existe.  
\* Verificamos que de acuerdo a las especificaciones del script no exista ya el fichero en la carpeta destino.  
\* Realizamos la copia. Redirigimos todo a nul para que no de mensajes de archivos copiados.  
\* Comprobamos el código de retorno y si no es cero enviamos el mensaje correspondiente.  
\* Damos un mensaje de copia correcta cuando se ha finalizado correctamente el script.

Bien, el ejemplo anterior nos puede servir para la introducción de lo que son los parámetros y cómo se deben usar o referenciar en el script. Vamos a ver toda la casuística más en detalle.

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. Hagamos una prueba:

**@echo off  
echo Parámetro 0: %0**

%1 a %9 son los posibles parámetros enviados en la línea de comandos. Realmente pueden pasarse más de 9 parámetros pero sólo 9 de ellos serán accesibles directamente. Si pasásemos más, hay que usar 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 (el %1 se pierde por lo que debemos guardarlo), el %3 pasa a ser el %2, etc... De tal manera que al final el %9 pasa a contener el posible décimo parámetro pasado. Realicemos como prueba:

@echo off  
echo Antes de desplazar  
echo Parametro 1: %1  
echo Parametro 2: %2  
echo Parametro 3: %3  
echo Parametro 4: %4  
echo Parametro 5: %5  
echo Parametro 6: %6  
echo Parametro 7: %7  
echo Parametro 8: %8  
echo Parametro 9: %9  
shift  
echo Despues de desplazar  
echo Parametro 1: %1  
echo Parametro 2: %2  
echo Parametro 3: %3  
echo Parametro 4: %4  
echo Parametro 5: %5  
echo Parametro 6: %6  
echo Parametro 7: %7  
echo Parametro 8: %8  
echo Parametro 9: %9  
shift  
echo Despues de desplazar de nuevo  
echo Parametro 1: %1  
echo Parametro 2: %2  
echo Parametro 3: %3  
echo Parametro 4: %4  
echo Parametro 5: %5  
echo Parametro 6: %6  
echo Parametro 7: %7  
echo Parametro 8: %8  
echo Parametro 9: %9

e invoquemos al script anterior (parm.cmd, por ejemplo) de la siguiente manera:

parm.cmd 1 2 3 4 5 6 7 8 9 10 11

Podemos referenciar "toda" la línea de parámetros mediante %\*

Si reescribimos el bat anterior:

**@echo off  
echo Toda la línea de parámetros es: %\***

y ejecutamos

parm.cmd 1 2 3 4 5 6 7 8 9 10

podemos comprobar la salida. Esto puede ser útil si luego queremos tratar todo como una sola línea y usamos comandos específicos, que veremos más adelante, para tratar dicha línea.

A los parámetros y sólo a los parámetros, pueden anteponérseles modificadores los cuales realizan una conversión especifica. Vamos a ver un ejemplo y a continuación daremos todas las posibilidades.

Hagamos un script (parm.cmd):

**@echo off  
echo Parámetro 1 expandido: %~f1  
echo Parámetro 2 expandido: %~f2**

Supongamos que nuestro script esté en la carpeta:

C:\ASIR1\W7\cmd>

Al ejecutar:

parm.cmd aaa.www bbb

nos devolverá:

Parámetro 1 expandido: C:\ASIR1\W7\cmd\aaa.www  
Parámetro 2 expandido: C:\ASIR1\W7\cmd\bbb

Es decir, el delimitador %~f por delante del numero de parámetro nos expande el parámetro considerándolo como si fuese nombre de archivo (sin verificarlo) al nombre completo de unidad, ruta y archivo en donde se esté ejecutando el script.

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.   
**%~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.   
**%~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.

La tabla siguiente enumera las combinaciones posibles de modificadores y calificadores que puede usar para obtener resultados compuestos.

**%~dp1** Expande %1 a una letra de unidad y una ruta de acceso.   
**%~nx1** Expande %1 a un nombre y extensión de archivo.   
**%~dp$PATH:1** Busca los directorios enumerados en la variable de entorno PATH para %1 y expande a la letra de unidad y ruta de acceso del primer directorio encontrado.   
**%~ftza1** Expande %1 a una línea de salida similar a dir.

En los ejemplos anteriores, se puede reemplazar %1 y PATH con otros valores de parámetros de proceso por lotes.

No se pueden manipular parámetros de proceso en los scripts de la misma manera en que se manipulan variables de entorno . No se pueden buscar y reemplazar valores ni examinar subcadenas. Sin embargo, se puede asignar el parámetro a una variable de entorno para, después, manipular la variable de entorno.

Aunque no es necesario memorizarlos sí que es importe saber que existen, ya que algunas veces nos puede ser necesario usarlos para expansión sobre todo en nombres de archivos.