COMANDOS ÚTILES

Comandos útiles para utilizar. Ejecuta **/?** después del comando para mayor información.

COLOR

```
C:\Users\Administrador>color 0a
```

Cambiar el color del primer plano y del fondo del CMD. Los argumentos son hexadecimales.

Escribe COLOR /? Para obtener una lista de códigos de color disponibles. ¡intenta otros!

ASSOC/FTYPE

```
C:\Users\Administrador>assoc .docx
C:\Users >ftype Word.Document.12
```

Permite ver o cambiar extensión de archivo asociada a cada programa (en esa PC).

TYPE

```
C:\Users\Administrador>type documento.txt
```

Permite mostrar el contenido del archivo sin lanzar un editor de texto.

CLIP

```
C:\Users\Administrador>dir /w | clip
C:\Users>clip < Diccionario.txt
```

Redirige la salida de un comando al portapapeles.

RUNAS

```
C:\Users\Administrador>runas /user:Administrador cmd.exe
```

Permite ejecutar un comando como otro usuario, generalmente como administrador.

DIR/COPY/XCOPY/MOVE/RENAME/DEL

DIR: Enumera el contenido de un directorio

COPY/XCOPY: Copiar archivos y carpetas.

MOVE: Mueve archivos y carpetas.

RENAME: Renombra archivos y carpetas.

PUSHD/POPD/CD/CHDIR/MD/MKDIR

PUSHD/POPD: Cambiar de directorio y guardar el actual.

CHDIR/CD: Cambiar del directorio actual.

MKDIR/MD: Crear nuevos directorios.

```
C:\> prompt $+%PROMPT%

C:\ >pushd c:\windows\system32
+C:\Windows\System32>popd
C:\ >
```

COMANDO ECHO

Por defecto el comando ***echo*** está habilitado, esto conlleva a que cada línea de un script batch sea mostrada y luego ejecutada. Por ejemplo:

Contenido del Archivo: ejemplo01.bat

```
echo Hola Esgeeks
```

Ejecución del archivo: ejemplo01.bat

```
C:\Users\Administrador> echo Hola Esgeeks
Hola Esgeeks
```

Puedes comprobar el estado actual de echo con el siguiente comando:

```
C:\Users\Administrador>@echo
ECHO está activado.
```

Por supuesto no queremos el ***echo*** habilitado. Hay dos métodos para deshabilitar ***echo***. Tomaré el ejemplo anterior.

```
@echo Hola Esgeeks
```

```
@echo off
echo Hola Esgeeks
```

Resultado de ejecutar el archivo ejemplo01.bat en ambos casos

```
C:\Users\Administrador>Hola Esgeeks
```

```
C:\Users\Administrador>Hola Esgeeks
```

Aprendamos rápido. El objetivo es colocar un salto de línea entre “3” y “Hola”; y juntar “Hola” con “Mundo” al final. A la derecha de muestra dos archivos correctos y se resalta con **negrita** el comando usado para el propósito.

```
C:\>ejemplo01.bat
1
2

Hola
Mundo

Hola Mundo
```

```
@echo off
echo 1
echo 2
echo.
echo Hola
echo Mundo
echo.
set /p "=Hola "<nul
set /p "=Mundo"<nul
```

```
@echo off
echo 1
echo 2
echo.
echo Hola
echo Mundo
echo.
<nul set /p "=Hola "
<nul set /p "=Mundo"
```

TRUCOS DE SALIDA

Es posible dividir un comando largo en múltiples líneas para mayor claridad. Si el símbolo "**^**" está presente al final de una línea, ignora que la siguiente línea es una continuación de la línea actual. (coloco el comando **pause** al final para que puedas ver la salida).

Contenido del Archivo: ejemplo02.bat

```
@echo off
echo Hola!
echo Gracias ^
por ^
comprar ^
esta ^
guía
echo Disfruta!
pause
```

Ejecución del archivo: ejemplo02.bat

```
Hola!
Gracias por comprar esta guía
Disfruta!
```

Es posible ejecutar múltiples comandos en la misma línea uniéndolos con el carácter "**&**":

Contenido del Archivo: ejemplo03.bat

```
@echo off
Echo Esto es una línea & Echo Esto es otra línea
pause
```

Ejecución del archivo: ejemplo03.bat

```
Esto es una línea
Esto es otra línea
```

Las declaraciones compuestas son tratadas como un solo bloque de comandos cuando están encerradas dentro del paréntesis "**( )**". Como puedes ver a continuación (izquierda), la consola te pedirá más comandos hasta que finalices el paréntesis.

```
C:\Users\EsGeeks>(echo 1
¿Más? echo 2
¿Más? echo 3
¿Más? )
1
2
3
```

En un script de archivo Batch, no verás el mensaje "¿Más?". Y lo crearás así:

```
@echo off
(
echo 1
echo 2
echo 3
)
```

Ahora intenta esto

```
@echo off
(
 (
  echo 1 & echo 2
  (
   echo Disfruta ^
de ^
este ^
truco
)
echo 3
)
)
```

C O M E N T A R I O S

Los comentarios permiten escribir un texto que será ignorado por el intérprete. Veamos las maneras de añadir comentarios en un archivo script BATCH.

1° forma: Usando la palabra clave "**REM**"

Contenido del Archivo: ejemplo04.bat

```
@echo off
REM esto es un comentario
REM esto es otro comentario
pause
```

2° forma: Usando la etiqueta ":" o "**::**"

Contenido del Archivo: ejemplo05.bat

```
@echo off
: esto es un comentario
:: esto es otro comentario
pause
```

Es posible añadir comentarios al final de un comando:

```
C:\Users\EsGeeks>ECHO Esto es un comando & :: Esto es un comentario
Esto es un comando
```

USO DE LA PALABRA CLAVE "**GOTO**" Y "**:**"

Es posible tener comentarios de varias líneas usando la palabra clave GOTO. El truco aquí es usar la palabra clave GOTO para saltar a una etiqueta más allá de los múltiples líneas que quieres que el intérprete ignore:

```
@echo off
ECHO MÚLTIPLES LÍNEAS COMENTADAS
GOTO despues_comentario
Hola
¡Esto es realmente divertido!
El intérprete no ejecutará estas líneas
:despues_comentario
ECHO FINAL MÚLTIPLES LÍNEAS COMENTADAS
pause
```

Salida de la ejecución del archivo

```
MÚLTIPLES LÍNEAS COMENTADAS
FINAL MÚLTIPLES LÍNEAS COMENTADAS
```

Entonces, cuando utilizamos GOTO todo lo que está después de esta palabra clave se omitirá, hasta llegar a la línea que contenga el ":" con el mismo nombre.

SÍMBOLOS ESPECIALES

Hay ciertos caracteres que tienen un significado especial:

- Caracteres de redireccionamiento de Entrada/Salida : "**<**" y "**>**".
- El carácter pipe: "**|**".
- El carácter caret: "**^**".
- El carácter de concatenación: "**&**".
- Otros caracteres como: "**%**" y "**!**"

Esos símbolos especiales tienen que ser "***escapados***" para poder ser usados. Para *escapar* de ellos, puedes anteponerles el carácter caret "**^**".

EJEMPLO: No puedes ejecutar el siguiente comando:

```
C:\Users\EsGeeks>ECHO 4 < 5, 3 > 2, 8 somos EsGeeks |, me divierto con esto ^
La sintaxis del comando no es correcta.
```

Para corregir la situación, antepondremos a cada uno de esos caracteres especiales el símbolo "**^**": **(Otra opción es rodear toda la cadena con comillas.)**

```
C:\Users\EsGeeks>ECHO 4 ^< 5, 3 ^> 2, 8 somos EsGeeks ^|, me divierto con esto ^^
4 < 5, 3 > 2, 8 somos EsGeeks |, me divierto con esto ^
```

Del mismo modo, si quieres imprimir el **signo de porcentaje** (que suele rodear una variable de entorno -ya lo veremos), puedes anteponerle otro signo de porcentaje como este:

```
@echo off
SET nombre=Alexynior
ECHO La variable %%nombre%% tiene el valor de %nombre%
pause
```

Salida de la ejecución del archivo

La variable %nombre% tiene el valor de Alexynior

Por último, el símbolo de exclamación "**!**" debe ser *escapado* también con un caret "**^**":

```
C:\Users\EsGeeks>ECHO Es el final de este truco^!
Es el final de este truco!
```

PASAR ARGUMENTOS

Los scripts Batch también pueden recibir argumentos de línea de comandos. Esto se logra prefijando el número de argumento con un signo de porcentaje ("**%**"). Los argumentos están separados por el carácter **espacial** a menos que estén rodeados por las comillas dobles.

Contenido del Archivo: ejemplo06.bat

```
@echo off
@echo arg0=%0, arg1=%1, arg2=%2
pause
```

Salida de: ejemplo06.bat “Es” “Geeks”

```
C:>ejemplo06.bat "Es" "Geeks"
arg0= ejemplo06.bat, arg1="Es", arg2="Geeks"
```

En lugar de obtener un argumento a la vez, es posible obtener todos los argumentos pasados usando el "**%***"

```
@echo off
@echo Todos argumentos: %*
pause
```

```
C:>archivo.bat Los argumentos son pasados
Todos argumentos: Los argumentos son
pasados
```

USO DE LA PALABRA CLAVE "**SHIFT**"

La palabra clave SHIFT desplaza todos los argumentos a la izquierda por uno. Cuando se pasan tres argumentos. Entonces el %1 original desaparece, el %2 se convierte en %1 y el %3 se convierte en %2, etc.

Archivo: ejemplo07.bat

```
@echo off
:REPETIR
if "%1"=="" goto FIN
echo Arg: %1
SHIFT
goto REPETIR
:FIN
pause
```

Salida de ejecución

```
C:>ejemplo07.bat 1 2 3 4
Arg: 1
Arg: 2
Arg: 3
Arg: 4
```

También puedes desplazar los argumentos en una posición determinada utilizando la siguiente sintaxis: SHIFT [/n]

INTENTA EJECUTARLO

```
@echo off
echo Args 1 a 4 son: %1 %2 %3 %4
shift /3
echo Nuevos Args 1 a 6 son: %1 %2 %3 %4
pause
```

V A R I A B L E S E N T O R N O

Cada programa tiene su propio bloque de entorno. Un bloque de entorno es sólo un almacén de memoria donde los programas pueden almacenar un conjunto de cadenas de la forma: "nombreVariable=valorVariable". En el lenguaje Batch, se utiliza la palabra clave **SET**.

Para **crear una nueva variable de entorno**, utiliza cualquier sintaxis de abajo:

```
SET nombreLibro=Batch con Esgeeks
```

```
SET nombreLibro=”Batch con Esgeeks”
```

La variable también puede ser numérica; y puedes crear una variable que utiliza otras variables:

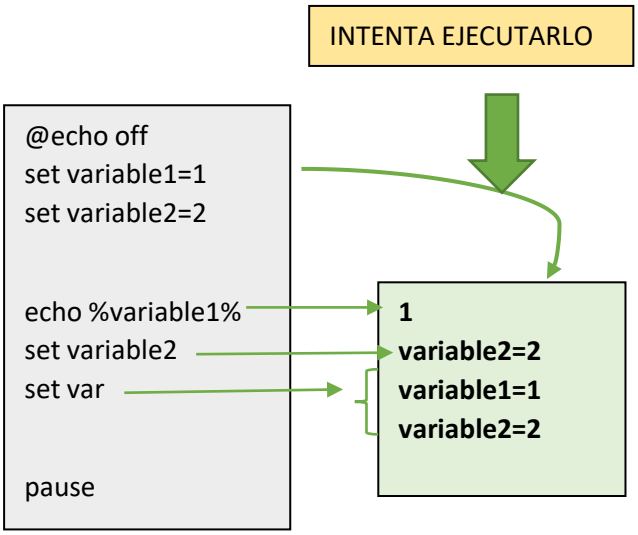
```
SET nivelLibro=2
```

```
SET nuevaVariable=El libro %nombreLibro%, es de nivel %nivelLibro%
```

Para borrar una variable de entorno, utiliza la palabra clave **SET**, pero no pases ningún valor después del signo igual ("="):

```
SET nombreVariable=
```

Para mostrar una variable de entorno, puedes utilizar la palabra clave **ECHO** (la variable entre signo de porcentaje), o simplemente utilizar la sintaxis "**SET nombreVariable**" (sin los signos de igual o de porcentaje).



Al usar la palabra clave **SET**, puedes usar el nombre completo de la variable de entorno o parte de su nombre.

SET var <-- ***muestra cualquier variable de entorno que coincida con el comodín "var*"***

SET nombreVariable <-- ***muestra el valor exacto de la variable de entorno que coincide con el valor***

V A R I A B L E S Ú T I L E S

Algunas variables de entorno están y predefinidas. Algunas de esas variables de entorno dinámicas pueden evaluarse a valores diferentes cada vez que se intentan explorar (por ejemplo el **%CD%**, **%TIME%**, **%RANDOM%**, etc.).

Establece la siguiente variable, y muestra su salida. Intenta también con la tabla más abajo.

```
@echo off
set aleatorio=%RANDOM%
set aleatorio
pause
```

INTENTA EJECUTARLO

aleatorio=18398

%CD%	Recupera el directorio actual.
%ERRORLEVEL%	Expande al código de salida de un comando.
%USERDOMAIN%	El nombre de dominio del usuario actualmente conectado.
%USERNAME%	El nombre de usuario actual.
%USERPROFILE%	Directorio de perfiles de usuario. (Descargas, Escritorio, etc.).
%APPDATA%	La carpeta de datos de la aplicación del usuario.
%windir% / %SystemRoot%	La carpeta de instalación actual de MS Windows.
%SystemDrive%	Letra de unidad de disco donde está instalado MS Windows.
%TMP% / %Temp%	Ubicación de la carpeta temporal de escritura.
%ProgramFiles%	La ubicación de los archivos del programa de 64 bits.
%PROCESSOR_ARCHITECTURE%	La actual arquitectura del procesador. x86 o AMD64.
%NUMBER_OF_PROCESSORS%	El número de procesadores virtuales.
%ComSpec%	Apunta al intérprete de comandos actual.
%PROMPT%	El valor actual del prefijo del prompt.
%DATE%	Muestra la fecha actual.
%TIME%	Muestra la hora actual.
%RANDOM%	Muestra un número decimal entre 0 y 32767.

```
@echo off
setlocal
set Nombre=Es Geeks
echo La variable %%Nombre%% no sera exportada al interprete...
endlocal
set Edad=25
echo La variable %%Edad%% si se exportara
pause
```

[Ejecuta lo siguiente desde la ubicación del archivo .bat de arriba]

```
set Nombre
set Edad
```

Podemos usar **SETLOCAL** y **ENDLOCAL** para que los cambios que se produzcan entre esas dos palabras clave no sobrevivirán al ámbito en el que se están ejecutando actualmente.

E T I Q U E T A S

En el lenguaje de scripting Batch, las etiquetas actúan como anclas o marcadores y se definen precediendo el nombre de la etiqueta con el carácter de dos puntos ("**:**"). Por ejemplo:

```
:MiEtiqueta
```

Para ir hacía una etiqueta, usa la palabra clave **GOTO**:

```
@echo off
echo Hola Geek
goto Etiqueta1
echo Se saltó esto
:Etiqueta1
echo Esto es la Etiqueta1
goto Etiqueta2
echo Se saltó esto
:Etiqueta2
Echo Esto es la Etiqueta2
pause
```

INTENTA EJECUTARLO

```
Hola Geek
Esto es la Etiqueta1
Esto es la Etiqueta2
Presione una tecla para continuar . . .
```

A menos que la palabra clave **GOTO** esté vinculada a otro comando condicional; intentar ir a una etiqueta inválida o inexistente hará que el script aborte con un error.

```
@echo off
goto %1
echo Código inalcanzable
:EtiquetaValida
echo Sí!
goto :eof
pause
```

El *código es inalcanzable* sin importar la etiqueta. Pero, si lo ejecutamos con un argumento de etiqueta válido:

```
D:\ESGEEKS >archivo.bat

El sistema no encuentra la etiqueta por lotes especificada:

D:\ESGEEKS >archivo.bat EtiquetaInvalida

El sistema no encuentra la etiqueta por lotes especificada: EtiquetaInvalida

D:\ESGEEKS >archivo.bat EtiquetaValida

Sí!
```

```
@echo off
goto %1 || (
echo ¡Falló en ir a esa etiqueta!
goto :eof
)

echo Código inalcanzable

:EtiquetaValida
echo EtiquetaValida alcanzada!
goto :eof
pause
```

Podemos hacer el script más resistente a los errores derivados de saltar a etiquetas inválidas o inexistentes uniendo la palabra clave **GOTO** con el lógico OR ("**||**"):

```
D:\ESGEEKS >archivo.bat

El sistema no encuentra la etiqueta por lotes especificada: ¡Falló en ir a esa etiqueta!
```

E T I Q U E A E O F

La etiqueta **":eof**" es una etiqueta especial y está disponible para su uso sólo cuando las Extensiones de Comando están habilitadas. Dependiendo del contexto, si saltas a esa etiqueta (con la palabra clave **GOTO**) entonces el script del archivo Batch terminará o volverá al *llamador*.

GOTO :eof

*No es lo mismo que **goto eof***

```
@echo off
goto eof
echo Inalcanzable, debido al salto de etiqueta
:eof
echo Una verdadera etiqueta está aquí
goto :eof
echo un código alcanzado, debido a "GOTO :EOF"
pause
```

```
D:\ESGEEKS >archivo.bat
Una verdadera etiqueta está aquí
```

Nota: Usar "**GOTO :EOF**" es equivalente a usar "**EXIT /B**" excepto que el primero no cambia el valor de la variable pseudoambiental **%ERRORLEVEL%.**

FUNCIÓN CALL

Puedes llamar a una etiqueta usando la palabra clave **CALL** precediendo el nombre de la etiqueta con los dos puntos (":"). Para volver de una "llamada a la función" sólo hay que emitir un "**GOTO :eof**".

```
@echo off
call :funcion1
call :funcion2 arg1 arg2
goto :eof

:funcion1
echo Esta es funcion 1
rem Regreso a la función
goto :eof

:funcion2
echo Dentro de la funcion 2
echo Los argumentos son: %*
goto :eof

pause
```

```
D:\ESGEEKS >archivo.bat
Esta es funcion 1
Dentro de la funcion 2
Los argumentos son: arg1 arg2
```

También puedes usar la sintaxis "**EXIT /B [retornoCodigoOpcional]**" para volver de una llamada de función.

T O M A R D A T O S

Es momento de hacer interactivo el script. Una forma de tomar una entrada de cadena del usuario se logra usando la sintaxis **SET /P**.

```
@echo off
set /P Nombre="Por favor, ingresa tu nombre: "
echo Bienvenido %Nombre%
pause
```

```
D:\ESGEEKS >archivo.bat

Por favor, ingresa tu nombre: Esgeeks
Bienvenido Esgeeks
Presione una tecla para continuar . . .
```

Supongamos que quieres seguir recibiendo información del usuario hasta que éste no introduzca nada con sólo pulsar la tecla ENTER:

```
@echo off
setlocal
:repetir
set entrada=
set /p entrada=Ingresa algo :
if "%entrada%"=="" goto :eof
echo -^> %entrada%
goto repetir
pause
```

```
D:\ESGEEKS >archivo.bat

Ingresa algo :4
-> 4
Ingresa algo :
```

Para este ejemplo, tienes que asegurarte de que la variable de ***entrada*** este vacía, antes de pasarla a "**SET /P**".

REDIRECCIONAR SALIDA

Cuando ejecutas un comando que produce algo a la salida estándar (**stdout**), puedes elegir redirigir la salida a otro lugar: a otro archivo o dispositivo. Un uso típico de la redirección de la salida en los scripts de archivos Batch es redirigir la salida al dispositivo **NUL** o a un archivo nuevo o existente

>	**Redirigir a un nuevo archivo.**
>>	Añadir a un archivo existente o crear un nuevo archivo (si el archivo no existía antes)

```
@echo off
ECHO Esto se escribirá en un archivo>archivo1.txt
ECHO Esto será agregado o escrito en el archivo2.txt>>archivo2.txt
ECHO Esto también se escribirá en el archivo2.txt >>archivo2.txt
ECHO Esto sobreescribirá archivo1.txt>archivo1.txt
ECHO ¡No se mostrará nada!>NUL
```

P I P E S

El concepto de pipes es muy similar al concepto de redireccionamiento de E/S, pero no son la misma cosa. Como su nombre sugiere, un pipe se utiliza para conectar dos programas juntos: la salida de un programa se canaliza y se utiliza como entrada en otro programa.

FIND, MORE, FINDSTR

Find *busca una cadena de texto en uno o más archivos.*

```
C:\EsGeeks>echo Esgeeks | find "geek"
Esgeeks
```

Esos dos comandos separados pueden ser canalizados juntos. La primera produce una salida y la segunda pide una entrada.

Otra utilidad es **MORE**. Es útil cuando se ejecuta un comando que produce una salida larga o cuando se está volcando el contenido de un archivo de texto (usando la palabra clave **TYPE**) y tú quieren leer la salida una página a la vez. INTENTA EJECUTAR LO SIGUIENTE:

```
DIR %WINDIR%
```

```
DIR %WINDIR% |MORE
```

El comando **MORE** tiene varios argumentos. Si se le pasa el argumento "+N", entonces MORE se saltará N líneas y comenzará a mostrar la salida después de N líneas:

```
linea 1
linea 2
linea 3
linea 4
```

A la izquierda, el contenido del archivo líneas.txt. A la derecha la ejecución y salida del comando.

```
D:\ESGEEKS >TYPE lineas.txt | MORE +1
linea 2
linea 3
linea 4
```

En el siguiente ejemplo simple, usaré el comando **FINDSTR** con el parámetro "**/V**" para mostrar todas las líneas excepto las que contienen la cadena (**/C**) "línea 3":

```
D:\ESGEEKS >TYPE lineas.txt  | findstr /V /C:"linea 3"
linea 1
linea 2
linea 4
```

OPERACIÓN ARITMÉTICA

Es posible hacer operaciones aritméticas en los scripts Batch. La palabra clave **SET** con el parámetro "**/A**" puede ser usada para ese propósito:

Algunas posibles operaciones:

+, -, *, / (división) y "%" (módulo)	Operaciones aritméticas.
~	Operador para acortar nombres largos de directorio.
!	Operador lógico de negación.
>>=, <<=, *=, +=, etc.	Operadores de asignación.
,	Separar expresiones múltiples usando el carácter de coma.

Aquí hay un ejemplo completo para ilustrar la mayoría de las operaciones: (a la izquierda el código en Batch; a la izquierda, el resultado de la ejecución)

```
@echo off
:: No se exponen todas las variables de entorno
SETLOCAL
SET /A var1="1+(2*4)"
SET /A var2=var1 + 2
:: Operador AND
SET /A var3="var1 & 1"
SET /A var4="var1 %% 2"
SET /A var5="1<<7"
:: Negación +/- 1
SET /A var6="~var1"
:: Número hexadecimal
SET /A var7="0x1234"
SET /A var8=1
SET /A var8^<^<=2
:: Mostrar todas las variables
SET var
IF %var3% EQU 1 echo var1 es impar
ENDLOCAL
```

```
var1=9
var2=11
var3=1
var4=1
var5=128
var6=-10
var7=4660
var8=4
var1 es impar
```

C O N D I C I O N A L E S : I F

Ningún lenguaje está completo sin permitir declaraciones condicionales. Aquí está la sintaxis básica de la palabra clave **IF**:

```
IF [NOT] ERRORLEVEL numero
IF [NOT] string1==string2
IF [NOT] EXIST nombre_archivo
```

*Después de cada condición **IF**, tienes que pasar una orden. Por ejemplo, para comprobar si un archivo determinado existe y luego llevar a cabo una acción después:*

```
IF EXIST archivo.bat ECHO ¡El archivo existe!
IF EXIST archivo.txt DEL archivo archivo.txt
IF NOT EXIST utilidad.exe ECHO La utilidad no fue encontrada! & GOTO fin
```

La palabra clave **IF** puede ser emparejada con la palabra clave **ELSE** que se utiliza para complementar lógicamente la condición:

```
IF (Comando) ELSE Comando
```

Fíjate en cómo rodeo el comando **IF** entre paréntesis (opcionalmente también podemos rodear el comando **ELSE** entre paréntesis).

```
IF EXIST archivo.txt (ECHO El archivo existe) ELSE ECHO El archivo no existe!
```

MÚLTIPLES COMANDOS

Es posible ejecutar múltiples comandos (o ***declaraciones compuestas***) dentro de una cláusula corporal de IF o ELSE. Es necesario abrir el paréntesis justo después de la condición en la misma línea que las palabras clave IF o ELSE:

```
IF condición (
Comando1
Comando2
…
)
```

```
IF condición (
Comando1
…
) ELSE (
Comando de Else
)
```

```
IF condición (
   IF condición (
…)
) ELSE (
   IF condición (
  ) ELSE (
  …
)
)
```

E S T R U C T U R A S C O N T R O L

La palabra clave **FOR** permite repetir la ejecución de un solo comando o de varios comandos para cada elemento en su parámetro "***set***".

Cuando se usa desde el intérprete de comandos directamente:

FOR %variable IN (**set**) **DO** comando [parámetros-comando]

Cuando se usa en un script de archivo Batch, debes anteponer a la variable el carácter de doble porcentaje ("**%%**")

FOR %%variable IN (**set**) **DO** comando [parámetros-comando]

Aquí hay algunos ejemplos de nombres de variables válidos que pueden ser usados con el bucle FOR dentro de tu script:

"%%a" a "%%z" y sus contrapartes mayúsculas **("%%A" a "%%Z"). "%%1" a "%%9"** y también: **"%%#", "%%@", etc.**

Los nombres de archivos en el parámetro "***set***" pueden contener el comodín "**?**" para que coincida con cualquier carácter individual; o "*****" para coincidir con cualquier carácter con cero o más ocurrencias.

```
@echo off
setlocal
set /a c=0
for %%a in (t*.bat,*.exe, *.txt) do (
 echo encontrado: %%a
 set /a c=c+1
)
echo.
echo Lista de %c% archivo(s)
endlocal
```

```
D:\ESGEEKS >archivo.bat
encontrado: COMUNIDAD.txt
encontrado: lineas.txt

Lista de 2 archivo(s)
```

En el ejemplo de la izquierda, se intenta listar los archivos con extensiones E.bat (en este caso, que empiece con E), .exe y .txt. En mi ejemplo, encontró dos archivos TXT.*

También puedes colocar una serie de cadenas para ser procesadas en un bucle: ¡Intenta ejecutarlo!

```
@echo off
setlocal
for %%a in (
"comando 1"
"comando 2"
"comando 3"
"cadena 1"
"cadena 2") DO (
ECHO elemento=%%~a
)
endlocalarchivo(s)
endlocal
```