

Windows Scripting (part 3)

11.- Sentències iteratives o repetitives dels shells scripts

L'interpret de comandes també reconeix la comanda **FOR** que permet executar una sèrie de comandes varies vegades. D'aquesta forma s'aconsegueix que el codi del shell script sigui molt més reduït en mesura, encara que el nombre d'instruccions que s'executa sigui elevat.

En una execució iterativa o repetitiva (també anomenada bucle), les comandes que s'executen són les mateixes, però s'apliquen sobre diferents dades. Si pensem, per exemple, en una operació d'eliminació d'un arxiu, no resulta lògic que aquesta es faci varies vegades sobre el mateix arxiu; en tot cas, sobre arxius diferents. Per aquesta raó, en un bucle FOR s'ha d'establir algun mecanisme per a que, cada cop que es torni a executar el mateix codi, canviï la informació que s'està gestionant.

També cal preparar algun mecanisme per a que el bucle deixi d'executar-se en algun moment. D'aquesta manera, s'evita que les instruccions que en formen part s'executin indefinidament.

Quan es crea un bucle utilitzant la comanda *FOR* aquesta pot realitzar un recorregut per actuar sobre les següents col·leccions de dades :

- Una o varies cadenes de caràcters
- La sortida que resulta de l'execució d'una comanda (en forma de varies línies de text)
- Un conjunt d'arxius o carpetes
- El contingut d'un arxiu de text
- Un rang numèric de valors amb un increment també numèric.

La sintaxis genèrica de la comanda FOR és la següent :

FOR /modificador %%variable IN (colecció) DO comanda args

El *modificador* és un argument que especifica el tipus de dades que gestionarà la comanda. Aquest modificador pot prendre qualsevol dels valors especificats en la taula que es veu a continuació.

Tabla Modificadores del comando *FOR*

Modificador	Tipo de datos
/D	Archivos que se encuentran en una determinada carpeta.
/F [opciones]	<p>Elementos de una cadena de caracteres. Si no se especifican opciones, se pasa como valor a la variable iterativa la cadena contenida en cada fila de texto hasta encontrar el primer espacio en blanco. Se puede cambiar este comportamiento seleccionando de forma distinta los elementos de cada fila de texto con las siguientes opciones:</p> <ul style="list-style-type: none"> ■ "DELIMS=c": establece el carácter c como delimitador de cada elemento de la cadena (por defecto es el espacio en blanco y la tabulación). ■ "EOL=c": establece el carácter c como el final de línea de la cadena, de forma que lo que sigue a continuación se ignora. ■ "SKIP=n": número de elementos que no se procesan (se saltan) desde el principio del texto. ■ "TOKENS=rango": rango de elementos a procesar (puede indicarse con comas o con un guión). Si se especifica más de un elemento (valor por defecto), entonces se crearán de forma implícita todas las variables iterativas a continuación de la variable iterativa declarada explícitamente en el bucle <i>FOR</i>.
/L	Rango de valores.
/R	Todas las carpetas que se encuentran dentro de una determinada carpeta.

Per la seva part, *variable* fa referència a una variable especial, anomenada **variable iterativa**, que va prenent cada valor de la colecció de dades conforme es repeteix la comanda. Per exemple, si la colecció de dades es un conjunt d'arxius que estan ubicats dins d'una carpeta, aquesta variable anirà prenent com a valor cadascun d'aquests arxius. Aquesta variable només pot nombrar-se amb un únic caràcter de la "a" a la "z" o de la "A" a la "Z" i no existeix fora de la sentència *FOR*.



NOTA

Hay que tener cuidado ya que, cuando se especifica un nombre de variable iterativa en el comando *FOR*, se distingue entre mayúsculas y minúsculas. Por eso, la variable %%A es distinta a la variable %%a.

El que s'ha especificat a *colecció* es refereix al conjunt de dades que es recorrerà en el bucle i que pot tractar-se d'una cadena de caràcters (delimitada entre cometes dobles), una llista d'arxius o carpetes (especificats sense cometes), la sortida d'una comanda (especificada entre cometes simples) o un

rang numèric (especificat per tres valors separats per comes : inici, increment i fi. Finalment, *comanda* especifica la comanda que s'executa cada cop que el bucle FOR dona una volta. Aquesta comanda pot tenir els seus propis arguments especificats a *args*.

Anem a veure a continuació uns quants exemples :



EJEMPLO

El siguiente código muestra por pantalla todos los archivos y carpetas que se encuentran en la carpeta actual:

```
@ECHO OFF
FOR /F %%a IN ('DIR /B') DO (
    ECHO %%a
)
```

Este bucle ejecuta el comando *DIR /B* que devuelve un texto formado por filas, donde cada una de ellas es el nombre de un archivo o carpeta. Cada vez que da una vuelta al bucle *FOR*, la variable iterativa *%%a* toma el valor de una de esas filas, puesto que los datos manejados son un conjunto de cadenas de caracteres.

Hay que tener cuidado ya que el comportamiento predeterminado de la opción */F* sin parámetros consiste en pasar a la variable iterativa el valor de la fila de texto hasta que encuentre el primer espacio en blanco. Por esta razón, en este código de ejemplo no se muestran correctamente los archivos y carpetas cuyos nombres contienen espacios en blanco.



EJEMPLO

El siguiente código muestra por pantalla todas las palabras que forman una frase almacenada en una variable de entorno:

```
@ECHO OFF
SET cadena=Dios^ no^ juega^ a^ los^ dados
FOR /F "TOKENS=1-5" %%a IN ("%cadena%") DO (
    ECHO %%a
    ECHO %%b
    ECHO %%c
    ECHO %%d
    ECHO %%e
)
```

Este bucle solamente se ejecuta una vez puesto que la cadena solamente tiene una línea de texto. Se ha utilizado el símbolo *^* delante de cada espacio en la asignación del valor de la variable de entorno para que los espacios en blanco sean interpretados correctamente. En el bucle *FOR* se ha especificado que se toman las cinco primeras palabras de la cadena, por lo que se declaran implícitamente las variables de iteración que van a continuación de la *a* (*b*, *c*, *d* y *e*).



EJEMPLO

El siguiente código muestra por pantalla todas las variables de entorno definidas en la línea de comandos:

```
@ECHO OFF
FOR /F "DELIMS== TOKENS=1" %%a IN ('SET') DO (
    ECHO %%a
)
```

Este bucle ejecuta el comando *SET* para establecer el conjunto de datos sobre el que va a trabajar. Este conjunto está formado por un bloque de texto en diferentes filas, donde cada una de ellas contiene una variable de entorno, el símbolo igual a y su valor. Se ha establecido como delimitador de columnas este símbolo igual (*DELIMS==*) y se indica que se toma solamente la columna uno (que coincide con el nombre de la variable). En la variable iterativa *%%a* se guarda el nombre de la variable, que se muestra en pantalla con el comando *ECHO*. Si deseamos mostrar también el valor de las variables de entorno, entonces tenemos que cambiar el código de la siguiente forma:



EJEMPLO

```
@ECHO OFF
FOR /F "DELIMS== TOKENS=1-2" %%a IN ('SET') DO (
    ECHO %%a
    ECHO %%b
)
```

En este caso, al indicar que se toman dos columnas de cada cadena (1-2), entonces el comando *FOR* crea de forma implícita la variable de entorno *%%b* que contiene el valor de cada variable (columna dos).



EJEMPLO

Supongamos que tenemos un archivo de texto llamado "claves" en la carpeta actual que contiene nombres de usuarios y contraseñas separados por dos puntos:

```
paqui:cordoba1
marisabel:pullopelao
eduardo:12345
clara:clarinete
```

Podemos tener el siguiente código que lee los nombres de usuario de este archivo y los muestra por pantalla:

```
@ECHO OFF
FOR /F "DELIMS=: TOKENS=1" %%a IN (claves) DO (
    ECHO %%a
)
```

Si el archivo que se lee no se encuentra en la carpeta actual, entonces hay que indicar su ruta absoluta o relativa.



EJEMPLO

El siguiente código muestra por pantalla todos los archivos de extensión EXE ubicados en la carpeta actual:

```
@ECHO OFF
FOR %%a IN (*.exe) DO (
    ECHO %%a
)
```



EJEMPLO

En este caso, el conjunto de datos está formado por todos los archivos de la carpeta actual que tienen extensión EXE "*.exe". Por cada iteración del bucle, la variable %%a va tomando como valor los nombres de los archivos que tienen esta extensión. Este *script* se podría modificar para eliminar todos estos archivos:

```
@ECHO OFF
FOR %%a IN (*.exe) DO (
    DEL /Q %%a
)
```

También se puede modificar el código para que muestre las carpetas que se encuentran dentro de la actual:

```
@ECHO OFF
FOR /D %%a IN (*) DO (
    ECHO %%a
)
```



EJEMPLO

El siguiente código muestra por pantalla los diez primeros números impares:

```
@ECHO OFF
FOR /L %%a IN (1,2,10) DO (
    ECHO %%a
)
```

En este caso, el conjunto de datos está formado por todos los números enteros empezando por el uno, incrementando de dos en dos, hasta el 10. La variable iterativa %%a va tomando los valores (1, 3, 5, 7 y 9) en cada vuelta del bucle.

A l'adreça <http://www.palomatica.info/juckar/microsoft/msdos/bat/for.html> trobareu una referència addicional molt útil en quant al funcionament (i exemples) associats al FOR.

12.- Altres sentències de canvi de fluxe d'execució en shell scripts

Com s'explica en apartats anteriors, els programes *shell scripts* estan formats per una sèrie de comandes normalment escrites en una línia, que s'executen de dalt a baix. Quan apareix una comanda condicional com ara IF o una comanda iterativa com ara FOR, llavors aquest fluxe d'execució de dalt a baix pot canviar, saltant algunes comandes sense executar-les o executant-les varies vegades.

En aquest apartat s'expliquen altres comandes que permeten canviar l'ordre d'execució de les comandes. Sempre existeix una execució des de dalt a baix, però ara veurem que existeixen comandes que indiquen a l'interpret que continui l'execució en un punt diferent del *script*. Cal tenir cura alhora d'utilitzar aquest tipus de comandes ja que, en programes complexos ens poden confondre i fer que perdem fàcilment l'ordre d'execució que volem establir.

Existeixen dues comandes que canvien l'ordre d'execució en un script :

- **GOTO** : s'utilitza per indicar que l'execució del programa continuarà en un altre punt. Aquesta comanda accepta com a paràmetre el punt on ha de continuar l'execució del *script*, també anomenat **etiqueta**, la qual ha d'haver sigut marcada amb un nom anteposant el símbol de dos punts ":". Existeix per defecte una etiqueta anomenada **EOF** que, si la cridem amb GOTO, ens porta al final de l'arxiu *script* i, per tant, acaba la seva execució.

- **CALL** : s'utilitza per a indicar que s'ha d'executar un bloc de comandes que es troben en un altre lloc (dins del mateix *script* o en un de diferent) i, un cop fet això, l'execució continua just després de la comanda següent a CALL. Aquesta

forma d'execució també rep el nom de **crida a procediment** o **subrutina**, que és un mètode que inclouen molts llenguatges de programació. Aquest mètode resulta molt útil quan hi ha un bloc de codi que es repeteix varies vegades al programa, de manera que només cal escriure'l un cop i fer les crides a ell a través de `CALL`. Si es vol acabar l'execució del bloc de codi de la subrutina abans que el fluxe d'execució arribi al final del bloc de codi de la subrutina llavors cal utilitzar `GOTO :EOF`. Quan el programa principal crida a una subrutina amb `CALL` aquesta rep les mateixes variables d'entorn que el programa principal. Qualsevol modificació que es faci en les variables d'entorn dins de la subrutina, aquesta té efectes també en el programa principal.



NOTA

Si se desea evitar que una llamada a un procedimiento modifique variables de entorno que después sean utilizadas en el flujo principal de ejecución del *script*, entonces se pueden utilizar los comandos `SETLOCAL` y `ENDLOCAL`.



NOTA

Utilizando el comando `GOTO` es muy fácil cometer el error de crear un bucle infinito, de forma que la ejecución del *script* nunca termina.



EJEMPLO

Tenemos el siguiente código:

```
@ECHO OFF
ECHO a
GOTO :e1
ECHO b
ECHO c
:e1
ECHO d
```

Este programa muestra en pantalla "a d" ya que el comando `GOTO` hace que se salte la ejecución de las dos sentencias `ECHO b` y `ECHO c`. Se ha definido la etiqueta con el nombre "e1". Tenemos que tener cuidado con los bucles infinitos y, para este ejemplo, podemos crear uno si añadimos el comando `GOTO :e1` al final del *script*.



EJEMPLO

Tenemos el siguiente código:

```
@ECHO OFF
CALL :codigo1
CALL :codigo2
ECHO a
:codigo1
ECHO b
GOTO :EOF
:codigo2
ECHO c
GOTO :EOF
ECHO d
```

El resultado de la ejecución de este *script* es "b c a b", ya que primero se llaman a la subrutinas *codigo1* y *codigo2*. Después continúa la ejecución hasta que se llega al primer *GOTO :EOF*, donde termina el programa.



EJEMPLO

Tenemos el siguiente *script* llamado *prueba.bat*:

```
@ECHO OFF
CALL codigo2.bat
ECHO a
```

También tenemos este *script* llamado *codigo2.bat*:

```
ECHO b
ECHO c
```

El resultado de la ejecución del *script* llamado *prueba.bat* es "b c a", ya que primero se llama a la subrutina del *script* *codigo2.bat*. Después continúa la ejecución normalmente en *prueba.bat*.

Quan es fa una crida **CALL** a una subrutina del *script* local o a un altre arxiu, es possible indicar, a més del nom de la subrutina o de l'arxiu, una sèrie d'arguments. Aquests arguments es passen de la mateixa manera que els arguments que es passen a les comandes de la línia de comandes. Cal tenir cura quan es gestionen aquests arguments, ja que es poden confondre els arguments passats al *script* principal amb els arguments passats a la subrutina. Com a norma general, els arguments passats al *script* no son accessibles des de la subrutina o arxiu cridat des de **CALL**, excepte el cas que siguin passats al seu torn com arguments.



EJEMPLO

Tenemos el siguiente *script* llamado *prueba.bat* que se ha ejecutado desde la línea de comandos con el argumento "ayuda":

```
@ECHO OFF
ECHO %1
CALL :codigo1 arg2
GOTO :EOF
:codigo1
ECHO %1
GOTO :EOF
```

En el código anterior, el primer comando *ECHO %1* mostrará por pantalla "ayuda", ya que el *script* se ha ejecutado con este argumento. Sin embargo, el segundo *ECHO %1* mostrará por pantalla "arg2" ya que la subrutina *codigo1* se ha llamado con este argumento.

Cal anar amb compte quan s'utilitzen comandes GOTO i CALL, ja que ens podem despistar en quant al fluxe d'execució del programa. Molts programadors experimentats refusen utilitzar la comanda GOTO, la qual existeix en molt llenguatges de programació, ja que consideren que produeix un codi que pot arribar a ser bastant il·legible i a més, trenca totalment el paradigma de la programació estructurada i modular.

En llenguatges de programació bastant arcaics, com ara el BASIC, s'utilitzava el GOTO de forma intensiva.

13.- Opcions de debug de shell scripts

L'exemple que es mostra en la imatge que es veu a continuació es una tècnica molt utilitzada per permetre la detecció d'errors en un shell script. Un altre mètode molt utilitzat consisteix en redirigir tota la informació sobre l'execució del *script* (missatges, valors de les variables d'entorn, errors, etc...) a un arxiu de text (registre o log). Si el *script* no funciona correctament llavors el programador o l'usuari pot examinar aquest arxiu per ajudar-lo a entendre que pot estar passant.



EJEMPLO

Una forma muy elegante y efectiva de mantener el control de los errores de diseño de un *script* consiste en definir un modo de ejecución especial del programa llamado **depuración** o **debug**. Si se activa este modo en la ejecución del *script*, entonces se muestran por pantalla mensajes que indican las operaciones que se están ejecutando, valores de variables de entorno, mensajes de error, etc. En caso contrario, el *script* oculta estos mensajes para hacer la ejecución más limpia desde el punto de vista de la salida por pantalla.

El modo de ejecución de depuración se puede realizar de la siguiente forma:

```
@ECHO OFF
IF /I "%1"==" /D" (
    SET traza=ECHO
) ELSE (
    SET traza=REM
)
%traza% Comienzo de la ejecución en modo depuración
%traza% variable de entorno x con el valor %x%
```

El código anterior establece la variable de entorno *traza* al valor "ECHO" si se pasa como primer argumento del *script* el modificador /D; en caso contrario, se le da el valor "REM". A partir de ese momento, se puede utilizar la variable de entorno *traza* como si fuera un comando *ECHO* o *REM*, de forma que los mensajes indicados solamente se mostrarán en pantalla cuando el *script* se ejecute con el modificador /D. Estos mensajes pueden incluir mensajes detallados, valores de variables de entorno y cualquier otra información que se considere adecuada para detectar errores.

PRÀCTICA WINDOWS SCRIPTING

Nota : implementeu tots aquests scripts en el cmd de la maquina virtual Windows 7.

1.- Escriu un script que demani a l'usuari un numero i una potencia (tots dos han de ser numeros enters). A continuacio s'ha de mostrar el resultat del numero elevat a la potencia.

Calculeu el numero m elevat a la potencia n com $m*m*...*m$ n vegades (es a dir, el numero m multiplicat a si mateix n vegades).

2.- Escriu un script en el que es mostri una endevinalla (la resposta es una unica paraula no composta, es delimita aixi per no donar lloc a possibles diferents respostes) i es donin 10 intents per encertar-la. Passats 10 intents ha de sortir el missatge "Has perdut". Si l'endevinalla s'encerta ha de sortir el missatge "Has guanyat".

3.- Escriu un script que crei en el directori actual un directori que tingui com a nom "dataDDMMAAAA-HHMM", on DD es el dia actual, MM el mes actual, AAAA es l'any actual, HH es l'hora actual i MM els minuts actuals.

4.- Si tenim la variable d'entorn *cadena* amb el valor "La verdadera esencia del ser humano es la bondad/P/S/AH" (inicialitzada amb SET cadena="La

verdadera esencia del ser humano es la bondad/P/S/AH”), indica quina es la sortida de l'execució de les següents comandes a Microsoft Windows :

- (a) C:\> ECHO %cadena:~14,15%
- (b) C:\> ECHO %cadena:verdadera=%
- (c) C:\> DIR %cadena:~-6,2%

5.- Crea un *script* en Windows que comprovi si existeix una determinada carpeta i, si es així, crei una carpeta dins d'ella. El nom d'aquestes dues carpetes ha de ser passat al *script* com arguments.

6.- Crea un script en Windows que implementi el joc del numero màgic. Aquest joc consisteix en endevinar un nombre generat a l'atzar en un rang determinat. Quan l'usuari introdueix un numero el sistema indica si el numero màgic es major, menor o igual. El joc s'acaba fins que el numero s'endevini. Quan el numero s'endevini cal mostrar la quantitat d'intents que s'han necessitat. Nota : per generar el numero màgic (aleatori) pots utilitzar la funció predefinida %random% que genera un nombre aleatori entre 0 i 32767.

7.- Crear un script en Windows que obtingui les dades de varis arxius en els quals el patró coincideixi amb “Lista_*.txt”. Aquests arxius contenen, per files, els alumnes admesos en diferents centres educatius (amb el seu nom, cognoms i DNI, separats per dos punts “:”). El script ha d'escriure un arxiu “Resultado.txt” que guardi les mateixes dades dels alumnes admesos afegint el nom del centre (s'obte del nom de l'arxiu, per exemple “Lista_EscuelaMaestria.txt” o “Lista_IESMolina.txt” o “Lista_Copernic.txt) però sense duplicats. Els duplicats es corresponen amb alumnes admesos en mes d'un centre (les seves dades apareixen en mes d'un arxiu), de manera que nomes la primera sol·licitud admesa es guardi a “Resultados.txt”, mentre que les altres es guardin a “Duplicados.txt”.

8.- Escriu un script de creació pròpia en el qual es vegi al màxim tot el que hem vist sobre Windows Scripting. Comenta degudament el script i explica en els mateixos comentaris que fa i per a que serveix el script.