Programación BATCH Avanzada.

Temario.
1.- Variables y Argumentos.
2.- Imprimir caracteres de comandos.
3.- Algunas otras variables de Entorno.
4.- IF
5.- FOR
6.- Funciones
7.- Includes
8.- Misc
9.- EOF

Variables y Argumentos

El manejo de variables en batch, es muy flexible., y este nos permite hacer desde operaciones matemáticas, hasta seleccionar ciertas partes de una variable, asi como reemplazar cadenas de texto, y obtener archivos.. y sus propiedades, la fecha, hora, generar numeros aleatorios, entre otros.

Los argumentos que recibe un batch, son recibidos de la siguiente forma:

batch argumento1 dos tres

hara que:
%0 = batch
%1 = argumento1
%2 = dos
%3 = tres

en %0 se guardara, el nombre del archivo.

Podemos borrar el contenido de un parametro con el comando shift:

Código:
```
@echo off
echo %0 %1 %2 %3
shift /1
echo %0 %1 %2 %3
```

al llamar:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>astring 123 456 789
astring 123 456 789
astring 456 789
```

shift borro, el primer argumento.

Tambien contamos con los siguientes modificadores para los archivos:

Código:
```
    %~f1        - regresa la ruta y el archivo de %1.
       %~d1        - regresa la letra de la unidad de %1.
       %~p1        - regresa solo la ruta del archivo %1.
       %~n1        - regresa solo el nombre de archivo %1.
       %~x1        - regresa solo la extension del archivo %1.
       %~s1        - regresa solo la ruta, con directorios, con nombres cortos del
archivo %1.
       %~a1        - regresa los atributos del archivo.
       %~t1        - regresa la hora/fecha del archivo %1
       %~z1        - regresa el tamaño del archivo %1.
```

por ejemplo:

en un directorio donde tenemos:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>dir
 El volumen de la unidad C no tiene etiqueta.
 El número de serie del volumen es: A057-553B

 Directorio de C:\Documents and Settings\Administrador\tempcurso

24/07/2006  12:25a      <DIR>          .
24/07/2006  12:25a      <DIR>          ..
24/07/2006  12:25a                   6 archivo.txt
               1 archivos              6 bytes
               2 dirs  401,715,161,088 bytes libres
```

este batch:

Código:
```
@echo off
echo Ruta al archivo: %~f1
echo Disco: %~d1
echo Solo ruta: %~p1
echo Nombre: %~n1
echo Extension: %~x1
echo Ruta Corta: %~s1
echo Atributos: %~a1
echo Fecha: %~t1
echo Tamaño: %~z1
```

llamandolo analiza.bat, saca el siguiente resultado:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>analiza archivo.txt
Ruta al archivo: C:\Documents and Settings\Administrador\tempcurso\archivo.txt
Disco: C:
Solo ruta: \Documents and Settings\Administrador\tempcurso\
Nombre: archivo
Extension: .txt
Ruta Corta: C:\DOCUME~1\ADMINI~1\TEMPCU~1\archivo.txt
Atributos: --a------
Fecha: 24/07/06 12:25a
Tama±o: 6
```

Tambien podemos usar varias propiedades, por ejemplo, este codigo:
Código:
```
@echo off

echo %~anxt1
```

saca este resultado:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>analiza archivo.txt
--a------ 24/07/06 12:25a archivo.txt
```

Si lo que recibimos no es un archivo, sino una cadena de texto, que contiene mas de una palabra, se debe poner entre comillas, algo asi:

astring "parametro de varias letras"

al obtenerlo, en %1, lo recibimos con comillas, pero si usamos:

Código:
```
@echo off
echo Con Comillas: %1
echo Sin Comillas: %~1
```

saca este resultado:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>astring "parametro de varias letras"
Con Comillas: "parametro de varias letras"
Sin Comillas: parametro de varias letras
```

Y por ejemplo, si queremos obtener todos los argumentos, se usa:

%*

es decir:

Código:
```
@echo off

echo Argumentos: %*
```

al ejecutar:

astring parametro de varias letras

nos regresa:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>astring parametro de varias letras
Argumentos: parametro de varias letras
```

Para asignar a una variable, el valor de algo escrito por el usuario, puedes usar:

set /P variable=

por ejemplo:
Código:
```
@echo off
echo ¿Como te llamas?
set /P miva=

echo Tu te llamas %miva%
```

hara algo asi:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>astring
¿Como te llamas?
Juan
Tu te llamas Juan
```

Ahora, si por ejemplo, queremos, hacer algo como, obtener las ultimas 3 letras del nombre:

Código:
```
@echo off
echo Escribe tu Nombre
set /P NOM=
echo %NOM:~-3%
```

Aqui lo que estamos haciendo en la ultima linea:

%NOM:~-3%
hara, obtener los ultimos 3 caracteres, como si se tratara de la funcion substring.

por ejemplo, esto:

%NOM:~1,3%

obtendra del segundo al cuarto caracter. (empiezas desde 0, recuerda)

Para reemplazar cadenas, dentro de otra variable, es la siguiente sintaxis:
Código:
```
@echo off
echo Escribe una frase
set /P FRA=
echo %FRA:e=XXX%
```

esto reemplazara las letras "e" por "XXX".

Para hacer calculos matematicos, se usa el modificado /A, de esta forma:

Código:
```
@echo off
set /A x=1
echo %x%
set /A x=x*9
echo %x%
```

tenemos las siguientes operaciones disponibles:

Código:
```
()                  - agrupar
* / %               - operadores aritméticos
+ -                 - operadores aritméticos
<< >>               - mayús lógica
&                   - AND
^                   - XOR
|                   - OR
= *= /= %= += -=    - asignación
  &= ^= |= <<= >>=
,                   - separador de expresión
```

Con este podemos usar numeros hexadecimales de la siguiente forma:

Código:
```
@echo off
set /A x=2
echo %x%
set /A x=x*0xff
echo %x%
```

al colocar **0x** estamos especificando, que a continuacion se pondra un valor hexadecimal.

Imprimir caracteres sin usar comandos

Ahora, se preguntaran, en las variabes, siempre usamos los signos %
Como imprimes uno?

veamos:

Código:
```
@echo off
set x=pru
set pru=HOLA

:: Esto imprimira el contenido de x
echo %x%

::Esto imprimira la letra x
echo x

::Esto imprimira %x%
echo %%x%%

::Esto imprimira el valor de x entre %%
echo %%%x%%%

:: Lo mismo
echo %pru%
echo %%pru%%
echo %%%pru%%%
```

En resumen, si quieren imprimir, el caracter %, deben colocarlo 2 veces.

Para otros caracteres, que ejecutan alguna accion en batch como:
&

|
<
>

debes colocar este caracter antes:^

por ejemplo:

Código:
```
echo <html><h1>Hola</h1></html> >index.html
```

no funcionara, pero:

Código:
```
echo ^<html^>^<h1^>Hola^</h1^>^</html^> >index.html
```

dara:
Código:
```
C:\Documents and Settings\Administrador\tempcurso>type index.html
<html><h1>Hola</h1></html>
```

Algunas otras variables de entorno

Tenemos otras variables de entorno que podrian servir, ademas de las comunes de ALLUSERSPROFILE y HOMEPATH, por ejemplo:

Código:
```
@echo off
echo Directorio Actual: %CD%
echo Fecha: %DATE%
echo Hora: %TIME%
echo Numero Aleatorio: %RANDOM%
```
dara como resultado:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>astring
Directorio Actual: C:\Documents and Settings\Administrador\tempcurso
Fecha: Lun 24/07/2006
Hora:  2:13:49.33
Numero Aleatorio: 24523

C:\Documents and Settings\Administrador\tempcurso>astring
Directorio Actual: C:\Documents and Settings\Administrador\tempcurso
Fecha: Lun 24/07/2006
Hora:  2:13:51.60
Numero Aleatorio: 3681
```

Otras instrucciones muy utiles, serian POPD y PUSHD, que sirven para guardar y regresar al directorio actual, por ejemplo:

Código:
```
@echo off
echo %CD%
PUSHD \
echo %CD%
POPD
echo %CD%
```

PUSHD funciona de la siguiente manera:

Guarda el directorio actual., y va a la ruta especificada.

POPD regresa al directorio guardado por PUSHD

podemos usarlos uno dentro de otro, asi:

Código:

```
@echo off
echo %CD%
PUSHD \
echo %CD%
PUSHD %homepath%
echo %CD%
POPD
echo %CD%
POPD
echo %CD%
```

IF

If, reconoce varias situaciones:

valores de error
comparar cadenas
existencia de archivos
comparaciones matematicas

como ya sabemos %ERRORLEVEL% almacena algunos valores de otras funciones.

al hacer

IF ERRORLEVEL 1 @echo ok ELSE @echo no

estariamos preguntando si ERRORLEVEL es 1, si lo es imprime OK, si no, imprime NO.

tambien podemos comparar cadenas, por ejemplo:

Código:
```
@echo off
IF "%~1"=="hola" echo hola
```

usamos %~1 para que aunque el usuario ponga comillas, no salga error

Código:
```
@echo off
IF /I "%~1"=="hola" echo hola
```

este codigo, solo responde si dices hola CON minusculas. /I es para hacer mas estricta la comparacion.

ahora, podemos usar IF en varias lineas:

Código:
```
@echo off
IF EXIST %~snx1 (
echo EXISTE!
) ELSE (
echo NO EXISTE
)
```

con parentesis ().

si vamos a usar comparaciones numericas, debemos usar los siguientes codigos de comparación:

Código:

```
    EQU - igual
    NEQ - no igual
    LSS - menor que
    LEQ - menor que o igual
    GTR - mayor que
    GEQ - mayor que o igual
```

por ejemplo:

Código:
```
@echo off
echo Cual es tu edad?
SET /P m=
IF %m% GEQ 18 (
echo ERES MAYOR DE EDAD
) ELSE (
echo NO ERES MAYOR DE EDAD
)
```

que regresa:

Código:
```
C:\Documents and Settings\Administrador\tempcurso>acon
Cual es tu edad?
19
ERES MAYOR DE EDAD

C:\Documents and Settings\Administrador\tempcurso>acon
Cual es tu edad?
17
NO ERES MAYOR DE EDAD
```

Tambien podemos saber si una variable ya fue definida:

Código:
```
IF DEFINED var (
echo SI
) ELSE (
echo NO
)
```

FOR

FOR es una herramienta muy completa, nos permite analizar segmentos de la salida de comandos, funciones y el contenido de archivos.

Tambien permite hacer bucles de la siguiente manera:

Código:
```
FOR /L %%var IN (inicio,incremento,fin) DO (acciones)
```

inicio, es el valor inicial, incremento es la cantidad a sumar, y fin es la cantidad a detenerse, por ejemplo:

Código:
```
FOR /L %%i IN (0,1,10) DO (echo %%i)
```
imprimira 0 1 2 3 4 5 6 7 8 9 10

sinembargo, la forma mas sencilla de usar FOR, es para numerar archivos, y hacer algo con ellos.. por ejemplo:

Código:
```
FOR %%x in (x*) DO echo %%x
```
mostrará todos los archivos que empiezan con x.

para SOLO mostrar directorios:
Código:
```
FOR /D %%x in (m*) DO echo %%x
```
el modificador "/D", mostrará solo los directorios, que empiesen con m.

El comando se puede hacer "recursivo", es decir, que se ejecute dentro de cada directorio, con el comando "/R", por ejemplo, en una estructura de directorios como la siguiente:

Código:
```
├─────adios
│    └─────algo
```

```
│       └───xx
└───hola
```

al ejecutar el comando:
Código:
```
FOR /R /D %%x in (a*) DO echo %%x
C:\adios
C:\adios\algo
```

sinembargo, al ejecutar el comando..
Código:
```
FOR /R %%x in (*x*) DO echo %%x
C:\ax.txt
C:\adios\sx.txt
C:\adios\wwx.txt
C:\adios\algo\kkx.txt
C:\adios\algo\xx\xx.txt
C:\hola\ax.txt
C:\hola\rx.txt
```

nos mostrara todos los archivos que coinciden con la sequencia especificada, incluso podriamos hacer un buscador en batch, con el uso de FINDSTR y FOR.

Tenemos otro modificador, /F que nos permite usar ciertas opciones para separar el resulado de las acciones.. comandos, cadenas, o archivos. Su sintaxis es la siguiente:

Código:
```
FOR /F ["opciones"] %var IN (conjunto) DO (acciones)
```

conjunto puede ser:
conjunto de archivos -> sin comillas
cadena de caracteres -> con comillas dobles (" ")
comando -> con comilla simple (' ')

las opciones son las siguientes:
eol -> todo lo que este despues de este caracter sera ignorado (para cada linea)
skip -> numero de lineas a saltarse al principio del archivo/comando.
delims -> esto sirve para separar las strings.. si no se coloca esta opcion, se usaran como separadores "espacio" y "tab"
tokens -> esto es para especificar cuales segementos, delimitados por "delims", seran pasados a las variables.. por ejemplo:
1,2,3
solo pasara los primeros 3 segmentos.
1,2*
pasara 2 segmentos, el primero y todo lo que este despues (el * regresa todo el resto de la linea)
1,2-6,8,9*
regresara 4 segmentos, el primero, desde el segundo hasta el sexto, el octavo y el resto de la linea, despues del noveno, el signo de menos (-) genera intervalos.

por ultimo, esta la opcion:
usebackq -> que cambia la forma de interpretar si es string o comando, de la siguiente manera:
'cadena'
`comando`

Nota: [`] es diferente al caracter de acento [´]

por ejemplo, el siguiente comando:

Código:
```
FOR /F "tokens=1,3-5,7-9,10* delims= " %%i IN ("George no es malvado, es bondadoso, siempre piensa en los demas.") DO echo %%i %%j %%k %%l tonto, %%m %%n %%o matar a %%p %%q
```

dara de resultado:
Código:
```
George es malvado, es tonto, siempre piensa en matar a los demas.
```

Funciones

El uso de argumentos, de etiquetas y de filtros nos ayuda mucho al momento de escribir un codigo.

Muchos creen que las etiquetas solo sirven para los goto.. sinembargo una etiqueta puede servir de funcion y recibir parametros.

miren, el siguiente programa:
Código:
```
@echo off
call:funcion 1 2 3
call:funcion %*
goto:EOF
:funcion
echo Estoy en: %~nx0-^>%0 %*
```

al ser llamado, por ejemplo en:
Código:
```
C:\>ejemplo
Estoy en: ejemplo.bat->:funcion 1 2 3
Estoy en: ejemplo.bat->:funcion
```
el primero muestra los parametros enviados por el batch (1 2 3), y el segundo los parametros enviados al programa.
en este otro ejemplo:
Código:
```
C:\>ejemplo HOLA MUNDO
Estoy en: ejemplo.bat->:funcion 1 2 3
Estoy en: ejemplo.bat->:funcion HOLA MUNDO
```
la funcion obtiene tambien los argumentos del programa.

hasta ahora.. todo es igual a usar goto a excepcion del uso de argumentos, sinembargo.. si queremos hacer un.. "return", se hace usando:
goto:EOF
entonces, en situaciones como:
Código:
```
@echo off&call:main&goto:EOF

:suma
set /A res=%1 + %2
echo %res%
goto:EOF

:resta
set /A res=%1 - %2
echo %res%
goto:EOF

:multiplica
set /A res=%1 * %2
echo %res%
goto:EOF

:main
set /P arg=Escribe 2 numeros separados por un espacio
echo %arg%
echo su suma es:
call:suma %arg%

echo su resta es:
call:resta %arg%

echo su producto es:
call:multiplica %arg%

goto:EOF
```

como podemos ver goto:EOF se usa para regresar al orden de comandos.

el resultado es algo asi:

Código:
```
C:\>operaciones
Escribe 2 numeros separados por un espacio 6 2
6 2
su suma es:
8
su resta es:
4
su producto es:
12
```

Includes
Para hacer un "include" oincluir un archivo, solo debes llamarlo asi:
archivo_a_incluir.bat funcion argumentos
y en el archivo a incluir, debe de estar al principio: **@echo off&call:%*&goto:EOF**

queda algo asi:
Código:
```
@echo off&call:%*&goto:EOF
:funcion1
...
:funcion2
...
```

por ejemplo:

--inclu.bat--
Código:
```
@echo off&call:%*&goto:EOF

:cabeza
echo ################
echo # Hecho por:  #
echo #     ¿?      #
echo ################
goto:EOF

:uso
echo uso:
echo %~nx0 Nombre
goto:EOF

:nombre
echo Hola %*
goto:EOF
```

--inicio.bat--
Código:
```
@echo off
if "%~1"=="" (
inclu.bat cabeza
inclu.bat uso
) else (
inclu.bat nombre %~1
)
```

esto daria este resultado:

Código:
```
C:\>inicio
################
# Hecho por:  #
#     ¿?      #
################
```

```
uso:
inclu.bat Nombre

C:\>inicio sirdarckcat
Hola sirdarckcat

C:\>
```

Misc
Algunos filtros y comandos interesantes:

- comando | more

muestra el resultado del comando de forma paginada

- comando | sort

muestra el resultado del comando de forma ordenada

- TITLE "algo"

cambia el titulo de la ventana de CMD

- SUBST ruta/a/alguna/carpeta [letra_unidad]

asgina a letra_unidad la ruta de acceso

- FIND "cadena"

busca cierta cadena en un archivo (se puede usar como filtro), con el modificador /v encuentra solo las que NO tienen la linea especificada. (para mas info, escribe en CMD: FIND /?)

- FINDSTR

extension de FIND, acepta expresiones regulares, y busqueda general (lo mas parecido que DOS tiene a grep), descripción detallada:
Código:

```
Busca cadenas en los archivos.

FINDSTR [/B] [/E] [/L] [/R] [/S] [/I] [/X] [/V] [/N] [/M] [/O] [/P]
[/F:archivo]
   [/C:cadena] [/G:archivo] [/D:lista de directorios] [/A:atributos de color]
   [cadenas] [[unidad:][ruta]nombredearchivo[ ...]]

  /B         Hace coincidir los modelos si están al principio de la línea.
  /E         Hace coincidir los modelos si están al final de la línea.
  /L         Literalmente utiliza cadenas de búsqueda.
  /R         Utiliza cadenas sde búsqueda como expresiones regulares.
  /S         Busca archivos que coinciden en el directorio actual y en todos
             los subdirectorios.
  /I         Especifica que la búsqueda no distingue mayúsculas de minúsculas.
  /X         Imprime líneas que coinciden con exactitud.
  /V         Sólo imprime líneas que no contienen una correspondencia.
  /N         Imprime el número de la línea antes de la línea que coincide.
  /M         Sólo imprime el nombre de archivo si el archivo contiene una
             correspondencia.
  /O         Imprime un carácter de desplazamiento antes de las líneas que
             coinciden.
  /P         Omite archivos con caracteres que no son imprimibles
  /A:attr    Especifica atributos de color con dos dígitos hexadecimales. Ver
             "color /?"
  /F:archivo  Lee la lista de archivos desde el archivo especificado
             (/ significa consola).
  /C:cadena   Utiliza una cadena especificada como una búsqueda de cadena
             literal.
  /G:archivo  Coje la búsqueda de archivos desde el archivo especificado
             (/ significa consola).
  /D:dir     Busca un signo de punto y coma de la lista delimitada de
             directorios
  cadenas    Texto que se debe buscar.
  [unidad:][ruta]nombredearchivo
             Especifica un archivo o  archivos que buscar.

Utiliza espacios para separar múltiples cadenas de búsqueda a no ser que
el argumento lleve un prefijo con /C.  Por ejemplo, 'FINDSTR "qué tal" x.y'
```

```
busca "qué" o "tal" en el archivo x.y.  'FINDSTR /C:"qué tal" x.y' busca
"qué tal" en el archivo x.y.

Expresión regular de referencia rápida:
  .         Comodín: cualquier carácter
  *         Repetir: cero o más ocurrencias de un carácter previo o de clase
  ^         Posición de línea: comienzo de la línea
  $         Posición de línea: fin de línea
  [clase]   Clase de carácter: cualquier carácter en la serie
  [^class]  Clase inversa: cualquier carácter que no esté en la serie
  [x-y]     Intervalo: cualquier carácter que esté dentro del intervalo
            especificado
  \x        Escape: uso literal de un metacarácter x
  \<xyz     Posición de palabra: principio de palabra
  xyz\>     Posición de palabra: fin de palabra

Para obtener una información más completa sobre expresiones regulares de
FINDSTR referirse al Comando de referencia Command en línea.
```

- start "titulo ventana nueva" [OPCIONES] comando argumentos

START tiene mas opciones de las que son usadas comunmente.
"titulo ventana nueva" especifica el titulo de la ventana que se generará.

las opciones entre otros contienen:
/Druta - el programa inicia en el directorio..
/B - la aplicación se inicia sin ventana
/I - se inicia la aplicacion con el entorno original, es decir las modificaciones al entrono hechas en esta sesion, no afectaran el nuevo programa
/MIN - La nueva ventana se inicia minimisada
/MAX - La nueva ventana se inicia maximisada
/SEPARATE - El programa se inicia en una zona de memoria separada
/SHARED - El programa se inicia en una zona de memoria compartida

Iniciar en prioridades:

/LOW - baja
/NORMAL - normal
/HIGH - alta
/REALTIME - tiempo real
/ABOVENORMAL - sobre lo normal
/BELOWNORMAL - debajo de lo normal

/WAIT - Inicia el programa, y espera hasta que termine de ejecutarse

por ejemplo, si quieren abrir un programa sin generar una ventana, por ejemplo netcat, podrian hacer algo asi:

start /B /SEPARATE /HIGH /I /D %WINDIR% nc -L -p 1337 |exit

que lo inicia en una zona de memoria separada, con prioridad alta, en el contexto original, y en %windir%

EOF
EOF es una etiqueta que define el fin del archivo, sirve para terminar funciones, y en este caso, solo sirve para terminar el documento  , espero les sirva