EBook Gratuito

APPRENDIMENTO batch-file

Free unaffiliated eBook created from
Stack Overflow contributors.

#batch-file

Sommario

Capitolo 1: Iniziare con il file batch

Osservazioni

Da Microsoft Technet:

> Con i file batch, chiamati anche programmi batch o script, è possibile semplificare le attività di routine o ripetitive. Un file batch è un file di testo non formattato che contiene uno o più comandi e ha un'estensione del nome di file .bat o .cmd. Quando si digita il nome file al prompt dei comandi, Cmd.exe esegue i comandi in modo sequenziale come appaiono nel file.

Nomi ed estensioni di file batch

Estensione	Osservazioni
.bat	Questa estensione viene eseguita con MS-DOS e tutte le versioni di Windows
.cmd	Utilizzato per file batch nella famiglia Windows NT
.btm	L'estensione utilizzata da 4DOS e 4NT

Per capire la differenza tra `.cmd` e `.bat` , vedi qui .

Evita i nomi che sono già il nome di comandi incorporati. come `tracert` . C'è un'utilità chiamata `tracert.exe` . Quindi, evita di nominare un file batch `tracert.bat`

Esecuzione di file batch

Il modo più semplice per eseguire un file batch è semplicemente facendo doppio clic sulla sua icona. Oppure incollare il percorso completo del file in un prompt dei comandi o semplicemente il suo nome se il comando Prompt è stato avviato dalla directory del file batch, quindi immettere.

Esempio:

```
C:\Foo\Bar>test.bat
C:\Foo\Bar>C:\Foo\Bar\Baz\test.bat
```

Examples

Aprire un prompt dei comandi

Il prompt dei comandi viene preinstallato su tutti i sistemi operativi Windows NT, Windows CE, OS / 2 ed eComStation ed esiste come `cmd.exe` , in genere in `C:\Windows\system32\cmd.exe`

Su Windows 7 i modi più veloci per aprire il prompt dei comandi sono:

Premi `Win` , digita "cmd" e quindi premi `Invio` .

- Premere `Win` + `R` , digitare "cmd", quindi premere `Invio` .
- Se hai una finestra di Explorer aperta, digita "cmd" nella barra degli indirizzi per aprire un prompt nella directory correntemente selezionata.
- Fai clic con il tasto destro del mouse su una cartella in Explorer tenendo `premuto Shift` e seleziona "Apri finestra di comando qui".

Può anche essere aperto navigando verso l'eseguibile e facendo doppio clic su di esso.

In alcuni casi potrebbe essere necessario eseguire `cmd` con autorizzazioni elevate, in questo caso fare clic con il tasto destro e selezionare "Esegui come amministratore". Questo può essere ottenuto anche premendo `Ctrl` + `Maiusc` + `Invio` invece di `Invio` quando si utilizza il modo 1 dei punti sopra.

Modifica e visualizzazione di file batch

Qualsiasi editor ASCII può modificare i file batch. Un elenco di editori che possono Sintassi Sintassi lotto evidenziazione può essere trovato qui . È inoltre possibile utilizzare il blocco note predefinito fornito con Windows per modificare e visualizzare un file batch, sebbene non offra l'evidenziazione della sintassi.

Per aprire il blocco note:

- Premi `Win` + `R` , digita il `notepad` e premi `Invio` .

In alternativa, il modo più "primitivo" per creare un file batch è reindirizzare l'output dalla riga di comando a un file, ad es.

```
echo echo hello world > first.bat
```

che scrive `echo hello world` al file `first.bat` .

È possibile modificare un file batch facendo clic con il pulsante destro sul file e selezionando "Modifica" dal menu di scelta rapida.

Per visualizzare il contenuto di un file batch da un prompt dei comandi, eseguire il seguente comando:

```
type first.bat
```

È anche possibile iniziare a modificare il file batch con il blocco note dal prompt dei comandi digitando

```
notepad first.bat
```

Ottenere aiuto

Per ottenere aiuto su un comando di file batch è possibile utilizzare la guida integrata.

Aprire un prompt dei comandi (il cui eseguibile è `cmd.exe`) e immettere l' `help` per visualizzare tutti i comandi disponibili.

Per ottenere aiuto per uno di questi comandi, digita `help` seguito dal nome del comando.

Per esempio:

```
help help
```

Mostrerà:

```
Provides help information for Windows commands.

HELP [command]

    command - displays help information on that command.
```

Alcuni comandi visualizzeranno anche help se seguiti da `/?` .

Provare:

```
help /?
```

Nota:

`Help` mostrerà solo l'aiuto per **i** comandi **interni** .

Leggi Iniziare con il file batch online: https://riptutorial.com/it/batch-file/topic/1277/iniziare-con-il-file-batch

Capitolo 2: Aggiungi ritardo al file batch

introduzione

Questo argomento ti insegnerà una delle tante cose utili da sapere nel linguaggio di scripting, file batch; Aggiunta di un ritardo / pausa / timeout al file batch.

Examples

Tempo scaduto

Tempo scaduto

Il modo più semplice per effettuare un ritardo o una pausa per un certo periodo di tempo, è con il comando standard `TIMEOUT` . Per fare un timeout che dura esattamente un minuto digitiamo:

```
timeout /t 60
```

Ora cosa sta succedendo qui?

Prima di tutto usiamo il comando `TIMEOUT` con il parametro `/T` (che significa semplicemente timeout), quindi specifichiamo la quantità di secondi da attendere. In questo caso ... `60` secondi.

Timeout con il parametro / NOBREAK

Se prendiamo l'esempio di prima e lo eseguiamo in un file BATCH: `timeout /t 60` poi, mentre attendi quei 60 secondi, sei effettivamente in grado di interrompere il timeout premendo un tasto qualsiasi sulla tastiera. Per evitare ciò, aggiungiamo semplicemente il parametro `/NOBREAK` alla fine di esso.

```
timeout /t 60 /nobreak
```

In questo modo si spegnerà per 60 secondi e, se si desidera interrompere il timeout, sarà necessario premere (CTRL-C) sulla tastiera.

Timeout silenzioso

Quando sta facendo un timeout mostrerà:

```
Waiting for X seconds, press a key to continue ...
or
Waiting for X seconds, press CTRL+C to quit ... [This is with the /NOBREAK parameter]
```

Per nascondere il messaggio usa l'argomento `NUL` (Per la spiegazione di `NUL` : **clicca qui**)

```
timeout /t 60 > nul
timeout /t 60 /nobreak > nul
```

Pausa

Per mettere in pausa lo script, usa semplicemente il comando `PAUSE` .

```
PAUSE
```

Questo mostrerà il testo `Press any key to continue . . .` , quindi aggiungi una nuova riga sull'input dell'utente.

Diciamo che vogliamo creare un programma "Hello World" e dopo aver fatto clic su qualcosa sulla nostra tastiera, vogliamo che esca dal programma con il comando `EXIT` .

```
echo Hello World
pause
exit
```

Qui usa il comando `ECHO` per dire "Hello World". Quindi usiamo il comando `PAUSE` che visualizza `Press any key to continue . . .` e quindi usiamo il comando `EXIT` per terminare lo script BATCH corrente.

Quando è in pausa, verrà visualizzato:

```
Press any key to continue . . .
```

Nascondere il messaggio "Premere un tasto qualsiasi per continuare ...

Per nascondere il messaggio, reindirizziamo l'output a un dispositivo speciale chiamato `nul` . Questo non è in realtà un vero dispositivo, ma qualunque cosa gli mandiamo viene buttato via.

```
pause > nul
```

ping

---

ping

Uno dei comandi più usati per ritardare per un certo periodo di tempo è il `ping` .

Utilizzo di base

```
PING -n 1 -w 1000 1.1.1.1

REM the -n 1 flag means to send 1 ping request.
REM the -w 1000 means when the IP(1.1.1.1) does not respond, go to the next command
```

```
PING 1.1.1.1 is an non-existing IP so the -w flag can ping a delay and go to next command
```

Ciò produrrebbe quanto segue sul tuo file batch / console:

```
C:\Foo\Bar\Baz>ping -n -w 1000 1.1.1.1

Pinging 1.1.1.1 (Using 32 bytes of data)
Request timed out

Ping statistics for 1.1.1.1
    Packets: Sent = 2,Received = 0, Lost = 1(100% loss)
```

Nascondi il testo echeggiato

Basta aggiungere `>nul` sul retro del comando per reindirizzare a null.

```
ping -n w 1000 1.1.1.1 >nul
```

Questo non produrrebbe nulla.

Dormire

Dormire

Sul vecchio sistema Windows, il `timeout` non è disponibile. Tuttavia, possiamo usare il comando `sleep` .

uso

```
sleep 1
```

Molto auto-esplicativo; dormi per 1 secondo. Tuttavia, `sleep` è un comando deperacted e deve essere sostituito dal timeout .

Disponibilità

Questo comando è disponibile sul vecchio sistema Windows. Anche `SLEEP.exe` è incluso nel Resource Kit 2003.

Per utilizzare `sleep.exe` , inserire il file eseguibile nella cartella `%Windir%\System32` . Quindi puoi usarlo come comando normale.

Leggi Aggiungi ritardo al file batch online: https://riptutorial.com/it/batch-file/topic/9123/aggiungi-ritardo-al-file-batch

Capitolo 3: Argomenti della riga di comando del file batch

Examples

Argomenti della riga di comando forniti per i file batch

Gli argomenti della riga di comando del file batch sono valori dei parametri inviati all'avvio del batch. Dovrebbero essere racchiusi tra virgolette se contengono spazi. In un file batch in esecuzione, gli argomenti vengono utilizzati per vari scopi, ad esempio il reindirizzamento a `:labels` , impostazione di variabili o comandi in esecuzione.

Gli argomenti sono indicati nel file batch utilizzando `%1, %2, ..., %9` .

```
@echo off
setlocal EnableDelayedExpansion
if not "%1"=="" (
    set "dir=%~1" & set "file=%~2"
    type !dir!\!file! | find /n /i "True" >nul^
     && echo Success! || echo Failure
)
exit /b

C:\Users\UserName> test.bat "C:\Temp\Test Results" "Latest.log"
Success!
```

Gli appunti:

- Nell'esempio precedente, le virgolette vengono rimosse utilizzando il modificatore argomento `%~1` .
- Le stringhe lunghe sono divise in più righe usando `^` , e c'è uno spazio prima del carattere nella riga successiva.

File batch con più di 9 argomenti

Quando vengono forniti più di 9 argomenti, è possibile utilizzare il comando `shift [/n]` , dove `/n` significa iniziare all'ennesimo argomento, n è compreso tra zero e otto.

Ripetizione degli argomenti:

```
:args
set /a "i+=1"
set arg!i!=%~1
call echo arg!i! = %%arg!i!%%
shift
goto :args
```

Nota, nell'esempio sopra riportato, la variabile di espansione `i` viene utilizzata per assegnare

valori di argomento alla matrice di variabili. Il comando `call` consente di visualizzare tali valori variabili all'interno del loop.

Argomenti di conteggio:

```
for %%i in (%*) do (set /a ArgCount+=1)
echo %ArgCount%
```

Imposta una variabile sull'argomento n:

```
set i=5
call set "path%i%=%%~i"
```

Spostamento degli argomenti tra parentesi

Consente di avere il seguente `example.bat` e chiamarlo con gli argomenti `1` , `2` e `3` :

```
@echo off

(
    shift
    shift
    echo %1
)
```

Poiché l'espansione della variabile cambierà dopo il raggiungimento del contesto delle parentesi finali, l'output sarà:

1

Poiché questo potrebbe essere un problema quando ti sposti tra parentesi per accedere all'argomento dovrai usare la chiamata:

```
@echo off

(
    shift
    shift
    call echo %%1
)
```

ora l'uscita sarà `3` . Dato che il comando `CALL` è usato (che porterà ad un'ulteriore espansione di variabili) con questa tecnica gli argomenti che accedono possono anche essere parametrizzati:

```
@echo off

set argument=1

    shift
    shift
    call echo %%%argument%
```

con espansione ritardata:

```
@echo off
setlocal enableDelayedExpansion
set argument=1

    shift
    shift
    call echo %%!argument!
```

l'output sarà

```
3
```

Leggi Argomenti della riga di comando del file batch online: https://riptutorial.com/it/batch-file/topic/4981/argomenti-della-riga-di-comando-del-file-batch

Capitolo 4: Batch e ibridi VBS

introduzione

`Batch` sono in grado di funzionare con funzionalità `VBS` aumentando ulteriormente la loro affidabilità. Ad esempio, `VBS` può gestire decimali, spazi e altre operazioni avanzate che non possono essere eseguite in `batch` . Inoltre è in grado di lavorare con oggetti WMI e ActiveX.

Examples

Esegui VBS con file temporanei

Il metodo old-school per l'esecuzione di un altro script da `batch` consiste nel `echo` lo script in un'altra posizione, quindi eseguirlo.

Questo metodo può essere rappresentato in questo modo:

```
@echo off
rem VBS below
    echo your vbs > TempVBS.vbs
    echo other vbs>>TempVBS.vbs
rem End of VBS

cscript //nologo TempVBS.vbs
del /f /s /q TempVBS.vbs
```

Il metodo sopra richiederebbe un sacco di `echo (vbs) >> TempVBS.vbs` , quindi ecco un modo per accorciarlo. *(codice di Aacini)*

```
@echo off
setlocal

rem Get the number of the "<resource>" line
for /F "delims=:" %%a in ('findstr /N "<resource>" "%~F0"') do set "start=%%a"

rem Skip such number of lines and show the rest of this file
(for /F "usebackq skip=%start% delims=" %%a in ("%~F0") do echo %%a) > Program.vbs

cscript //nologo Program.vbs
del /f /s /q Program.vbs
exit /b

<resource>
your vbs
another line of vbs
```

L'ultimo metodo è utilizzando gli `streams` . Un file può avere alcuni flussi. E ogni stream può contenere informazioni diverse.

```
@echo off
```

```
    echo vbs >%0:stream1
    rem This command redirect "vbs" into the stream1 of this script, then we can call it later

cscript %0:stream1 //nologo
   rem if you like, you can clear the stream1 of this file by:
   type nul>%0:stream1
```

Incorpora il codice vbscript in un file batch senza utilizzare i file temporanei

Ecco un esempio con la tecnica (hack) inventata dall'utente Liviu del forum dostips :

```
@echo off
echo Printed by CMD.EXE
cscript //nologo "%~f0?.wsf" //job:JS //job:VBS

exit /b %errorlevel%

----END OF BATCH CODE---
<package>
  <job id="JS">
    <script language="VBScript">

        WScript.Echo("Printed by VBScript"):

    </script>
  </job>
  <job id="VBS">
    <script language="JScript">

        WScript.Echo("Printed by JScript");

    </script>
  </job>
</package>
```

Poiché l'esecuzione del file `wsf` con host di script di Windows è sensibile alle estensioni, è possibile eseguire un file con qualsiasi estensione aggiungendo `?.wsf` alla fine del file (che è il nocciolo dell'hack). Mentre l'esempio di Liviu è probabilmente più robusto, il codice sopra è una versione più semplificata. Siccome wsh non si preoccupa molto delle cose al di fuori del nodo `<package>` non sei obbligato a mettere tutto nei commenti xml. Anche se bisogna fare attenzione ai simboli di reindirizzamento (`<` e `>`)

Leggi Batch e ibridi VBS online: https://riptutorial.com/it/batch-file/topic/10061/batch-e-ibridi-vbs

Capitolo 5: Bug nel processore cmd.exe

introduzione

Questo argomento si concentrerà sugli errori causati dai bug del processore. Ecco le cose che ci concentreremmo sulla causa e sulla soluzione del problema.

Osservazioni

Nell'esempio DEL File Extension , utente X. Liu nota che questo errore **non si** verifica quando l'estensione del file nel comando `DEL` è inferiore a 3 caratteri.

Examples

Confusione tra parentesi

Da questo sito web, l'OP ha notato un problema.

Causa

Considera il seguente frammento di codice.

```
if 1==1 (
    set /a result = 2*(3+4)
)
```

A prima vista, potresti pensare che `CMD.exe` lo elaborerebbe in questo modo:

- La condizione è vera, esegue il blocco di codice
- Imposta il valore del risultato variabile su `14`
- Continua

Tuttavia, `CMD.exe` questo modo:

- La condizione è vera, esegue il blocco di codice
- **Calcola `2*(3+4` , il `)` dopo che 4 viene elaborato alla fine del blocco di codice `if`**
- A random `)` è apparso!

Il secondo passaggio restituirà l'errore di `Unbalanced parentheses` .

Soluzione

Secondo un `set /?` di CMD.exe tedesco `set /?` , dovremmo citare le operazioni aritmetiche. Ecco un esempio.

Precedente	Risultato
`set /a result=2+5*4`	`set /a result="2+5*4"`

A proposito, secondo un `set /?` CMD.exe inglese `set /?` , le virgolette sono obbligatorie se gli operatori logici o di modulo sono presenti nell'espressione (anche se questo non è un passaggio *obbligato*).

Carattere di fuga improprio

In questa domanda di overflow dello stack , txtechhelp utente ha riscontrato un problema con il carattere `^` che potrebbe causare un problema di sicurezza.

Causa

```
anyInvaildCommand ^
```

Nota: assicurati che il segno di omissione (`^`) sia l'ultimo carattere! Qualsiasi `CR\LF` più non funzionerà affatto!

Il caret cerca il prossimo personaggio di fuggire. Tuttavia, non ci sono più personaggi disponibili per scappare, quindi `cmd` loop infinitamente, cercando un personaggio da scappare. In questo processo "loop", `cmd.exe` consumerà la memoria del tuo computer. E gradualmente mangiando tutta la memoria, mettendo il computer in ginocchio.

Questo problema può portare a problemi di sicurezza più seri in quanto uno potrebbe semplicemente inserire il codice nel proprio computer sbloccato.

soluzioni

- Utilizzare la codepage **UTF-16** potrebbe risolvere questo problema. Solo l' **UTF-8** o l' **ASCII** causerebbero l'errore.
- Assicurati che ci sia un `CR\LF` in più nel file, o semplicemente non usi il caret alla fine del file.

Extra

Questo bug sembra essere risolto in Windows 10.

Estensione del file DEL

Questo bug è stato segnalato da steve2916 da questa discussione sul forum di Microsoft Windows.

Causa

Considera una cartella con tali file.

```
TestA.doc
TestB.doc
TestC.docx
TestD.docx
```

Se vogliamo rimuovere tutto il file `.doc` in questa directory, di solito faremmo:

```
del *.doc
```

Tuttavia, questo comando rimuove anche i file `.docx` . Lo stesso accade sulle estensioni di file con questo modello.

File A	File B
Qualche nome. **abc**	Un altro nome. **abc** d

Come possiamo vedere, finché la stringa di estensione del file contiene la stringa utilizzata nel comando `del` , il file verrà eliminato. Si noti che questo errore si verifica solo quando la stringa di estensione utilizzata nel comando `del` ha almeno tre caratteri. Ad esempio, `del *.do` non elimina `A.doc` o `A.docx` .

Soluzione

Nel thread, l'utente MicroCompsUnltd ha notato che l'utilizzo di `Powershell` avrebbe risolto il problema.

Leggi Bug nel processore cmd.exe online: https://riptutorial.com/it/batch-file/topic/10694/bug-nel-processore-cmd-exe

Capitolo 6: Bypassare le limitazioni aritmetiche nei file batch

introduzione

I file batch consentono solo calcoli integer a 32 bit, sebbene questo possa essere bypassato con approcci diversi.

Examples

Utilizzando PowerShell

Poiché PowerShell viene installato di default su ogni sistema Windows dal 7/2008 in poi può essere utilizzato per calcoli più complessi:

```
@echo off
set "expression=(2+3)*10/1000"
for /f %%# in ('"powershell %expression%"') do set result=%%#
echo %result%
```

Presta attenzione alle doppie virgolette addizionali in `for /f` che evitano che le parentesi siano in conflitto con la sintassi del comando for.

Il problema potenziale è che PowerShell è molto più lento rispetto all'utilizzo di wsh / vbscript / jscript a causa del caricamento del .net framerwork

Usando jscript

`WSH/JScript` è installato su ogni sistema Windows poiché NT, quindi utilizzarlo per calcoli più complessi, lo rende piuttosto portatile. JScript è più facile da combinare con il file batch:

```
@if (@codesection==@batch) @then
@echo off

set "expression=2*(2+3)/1000"
for /f %%# in ('cscript //nologo //e:jscript "%~f0" "%expression%"') do set
result=%%#
echo %result%
:: more batch code

exit /b %errorlevel%
@end
WScript.Echo(eval(WScript.Arguments(0)));
```

Con questo approccio è possibile inserire l'intero codice in un unico file. È più veloce rispetto all'utilizzo di PowerShell. Qui e qui possono essere trovati script più avanzati (che possono essere usati come file esterni).

Emulazione di calcoli di carta e penna, implementazioni di funzioni matematiche

1. ***Qui*** può essere trovata la libreria matematica più completa che emula calcoli di carta e penna e consente di lavorare con numeri più grandi.
2. Ecco altri esempi di emulazioni penna e carta: ADD , Comparison , Multiply
3. Alcune implementazioni di funzioni matematiche possono essere trovate qui .

Leggi Bypassare le limitazioni aritmetiche nei file batch online: https://riptutorial.com/it/batch-file/topic/10702/bypassare-le-limitazioni-aritmetiche-nei-file-batch

Capitolo 7: Cambiare le directory e elencarne i contenuti

Sintassi

- echo% cd% - visualizza il percorso corrente della directory
- cd "C: \ path \ to \ some \ directory" -cambia il percorso della directory
- cd "% variable_containing_directory_path%" - cambia anche il percorso della directory
- cd / d E: - cambia in E: drive da una diversa unità
- cd / - cambia la directory di nuovo nell'unità corrente
- echo `%__CD__%` - visualizza il percorso corrente della directory con barra rovesciata finale (non documentata)
- echo% = C:% - La directory corrente dell'unità C: (non documentata)
- echo% = D:% - La directory corrente dell'unità D: se l'unità D: è stata letta nella sessione CMD corrente (non documentata)

Osservazioni

Perché è importante e quali sono i loro usi e vantaggi:

- per aprire file o applicazioni in una directory utilizzando batch
- per creare e scrivere e leggere i file in una directory utilizzando batch
- per conoscere ed elencare tutte le cartelle
- per sapere dove è in esecuzione il file batch

Examples

Per visualizzare la directory corrente

Formato e utilizzo:

```
echo %cd%
```

`%cd%` è una variabile di sistema che contiene il percorso della directory corrente

Per cambiare la directory corrente (senza cambiare unità)

Formato:

```
cd "<path>"
```

Esempio:

```
cd "C:\Program Files (x86)\Microsoft Office"
```

`cd` è un'abbreviazione di `chdir` e i due comandi si comportano nello stesso modo. Per motivi di coerenza, il `cd` verrà utilizzato in questo argomento.

Per navigare nella directory di un livello sopra la directory corrente, specificare la directory di sistema `..`

```
cd ..
```

Per spostarsi in una directory che si trova all'interno della directory corrente, è sufficiente `cd` al nome della cartella senza digitare il percorso completo (avvolgendo il nome della directory tra virgolette se contiene spazi).

Ad esempio, per immettere `C:\Program Files (x86)\Microsoft Office` nella directory `C:\Program Files (x86)` , è possibile utilizzare la seguente sintassi:

```
cd "Microsoft Office"
```

o

```
cd "C:\Program Files (x86)\Microsoft Office"
```

Navigazione verso una directory su un'unità diversa

`cd` da solo non consentirà a un utente di spostarsi tra le unità. Per passare a un'unità diversa, è necessario specificare l'opzione `/d` .

Ad esempio, passando da `C:\Users\jdoe\Desktop` a `D:\Office Work`

```
cd /d "D:\Office Work"
```

Come mostrare tutte le cartelle e i file in una directory

Utilizzo per elencare tutte le cartelle e i file nella directory corrente: `dir`

È inoltre possibile specificare una directory di destinazione: `dir C:\TargetPath`

Quando si specifica un percorso con spazi, deve essere racchiuso tra virgolette: `dir "C:\Path With Spaces"`

Modifica del drive senza CD / D

```
Pushd "D:\Foo"
dir
Popd
```

`Pushd` cambierà la directory nella directory seguente (in questo caso D: \ Foo. `Popd` ritorna alla directory originale.

Per cambiare la directory corrente nella root dell'unità corrente

Formato:

```
cd/
```

`cd/` è impostato per riportare la directory corrente alla radice dell'unità corrente

Leggi Cambiare le directory e elencarne i contenuti online: https://riptutorial.com/it/batch-file/topic/7132/cambiare-le-directory-e-elencarne-i-contenuti

Capitolo 8: Casuale nei file batch

Examples

Numeri casuali

Usando la variabile dinamica `%Random%` , possiamo ottenere un numero intero casuale compreso tra 0 e 32767. Ad esempio:

```
echo %random%
```

Ovviamente, restituisce un numero intero compreso tra 0 e 32767. Ma a volte vogliamo che si trovi in un intervallo specifico, ad esempio da 1 a 100.

Generazione di numeri casuali all'interno di un intervallo specifico

Il metodo di base per farlo è elencato di seguito.

```
set /a result=(%RANDOM%*max/32768)+min
```

dove `max` è il numero più alto che può essere generato e `min` è il numero più piccolo che può essere generato. Si noti che non si otterranno numeri decimali perché si `set /a` arrotondamento verso il basso automaticamente. Per generare un numero casuale decimale, prova questo:

```
set /a whole=(%RANDOM%*max/32768)+min
set /a decimal=(%RANDOM%*max/32768)+min
echo %whole%.%decimal%
```

Generare numeri casuali superiori a 32767

Se provi

```
set /a whole=(%RANDOM%*65536/32768)+1
```

molto probabilmente otterrai numeri casuali che sono dispari.

Per generare numeri maggiori di 32767, ecco un metodo migliore.

```
set /a result=%random:~-1%%random:~-1%%random:~-1%%random:~-1%%random:~-1%%random:~-1%
```

Il codice precedente estrae il 1 carattere da ogni `%random%` . Ma questo è fatto apposta.

Poi é il numero `random` potrebbe essere un numero di cifre, l'estrazione delle ultime 2 cifre non funzionerà. Ecco perché estraiamo solo l'ultimo carattere. In questo caso, abbiamo il 6 `%random:~-1%` , generando il massimo di `999999` e il minimo a `000000` , potrebbe essere necessario regolarlo in base alle proprie esigenze.

pseudocasuale

`cmd.exe` genera il seme in base al momento in cui è iniziata la sezione `cmd` , quindi se si avvia la sezione mutliple quasi alla stessa ora, il risultato potrebbe non essere abbastanza "casuale".

Alfabeti casuali

Sfortunatamente, batch non ha un metodo integrato per generare alfabeti, ma usando `%random%` e `for` loop, possiamo 'generare' alfabeti.

Questa è una semplice idea di come funziona.

```
set /a result=(%random%*26/32768)+1
for /f "tokens=%result%" %%I in ("A B C D E F G H I J K L M N O P Q R S T U V W X Y Z") do (
    echo %%I
)
```

- Il primo `set /a` istruzione genera un numero casuale `N` compreso tra 1 e 26
- L'istruzione `for /f` seleziona il `N` ° elemento da un elenco di A a Z.
    - Restituisce il risultato

È possibile inserire un totale di `31` articoli in un ciclo `for` 1, e oggetti praticamente illimitati usando [questo metodo]. (Ordine dei parametri del ciclo Batch - for

Pseudocasuale e uniforme casuale nel lotto

Distribuzione pseudocasuale

Accertando a questa risposta di Overflow dello stack , l'utente CherryDT ha sottolineato questo codice:

```
set /a num=%random% %% 100
```

non dà una distribuzione uniforme.

La variabile dinamica interna `%random%` **non** dà una **distribuzione uniforme,** ma il codice di cui sopra non sarà un casuale uniforme. Questo codice genera un numero casuale compreso tra 0 e 99, ma il risultato non sarà uniforme. 0 ~ 67 si verificherà più di 68 ~ 99 dal `32767 MOD 100 = 67` .

Per generare un casuale distribuito uniforme usando il codice sopra, allora `100` deve essere cambiato. Ecco un metodo per ottenere un numero che crea una distribuzione uniforme.

```
32767 mod (32767 / n)
```

dove `n` è un numero intero compreso tra 0 ~ 32767, il risultato può essere decimale e potrebbe non funzionare in batch.

Distribuzione uniforme

```
set /a result=(%RANDOM%*100/32768)+1
```

Questo metodo genererà una distribuzione uniforme. Evita l'uso di `%` , che è più simile a "resto" e quindi a "modulo" in uno script batch. Senza usare `%` , il risultato sarà uniforme.

In alternativa, ecco un metodo inefficiente, ma uniforme.

```
set /a test=%random%

if %test% geq [yourMinNumber] (
    if %test% leq [yourMaxNumber] (

        rem do something with your random number that is in the range.

   )
)
```

Modificare `[yourMinNumber]` e `[yourMaxNumber]` base ai propri valori.

Leggi Casuale nei file batch online: https://riptutorial.com/it/batch-file/topic/10841/casuale-nei-file-batch

Capitolo 9: Cerca stringhe in batch

Examples

Ricerca di stringhe di base

Il comando FIND può scansionare file di grandi dimensioni riga per riga per trovare una determinata stringa. Non supporta i caratteri jolly nella stringa di ricerca.

```
find /i "Completed" "%userprofile%\Downloads\*.log" >> %targetdir%\tested.log

TYPE scan2.txt | FIND "Failed" /c && echo Scan failed || echo Scan Succeeded
```

Il comando FINDSTR ha più funzioni disponibili e supporta la ricerca di espressioni regolari (REGEX) con caratteri jolly nella stringa di ricerca.

```
FINDSTR /L /C:"Completed" Results.txt

echo %%G | findstr /r /b /c:"[ ]*staff.*" >nul && echo Found!
```

Vedi le fonti di aiuto FIND e FINDSTR per maggiori informazioni.

Utilizzando i risultati di ricerca

Lo script seguente mostra una tecnica di *file diviso* più avanzata, in cui la funzione FOR scorre un elenco di file in una directory e ogni contenuto del file viene convogliato in FINDSTR che cerca una stringa contenente una sottostringa `var` preceduta da un numero indefinito di spazi e sostituita da qualsiasi altro testo. Una volta trovato, il file cercato viene sostituito con uno nuovo che contiene solo la porzione di testo sopra la stringa di ricerca.

```
@echo off
setlocal enabledelayedexpansion
pushd "%temp%\Test"
for %%G in ("*.txt") do (set "break="
    (for /f "tokens=*" %%H in (%%~G) do (
        if not defined break (
            echo %%H | findstr /r /b /c:"[ ]*var.*" >nul && set break=TRUE || echo %%H )
    )) >> %%~nG_mod.txt
    del %%~G & ren %%~nG_mod.txt %%G )
popd
exit /b
```

Nota, come l' `break=TRUE` impostazione `break=TRUE` consente di uscire dal ciclo FOR dal ciclo cercato, una volta trovata la prima occorrenza della stringa di ricerca.

Leggi Cerca stringhe in batch online: https://riptutorial.com/it/batch-file/topic/5476/cerca-stringhe-in-batch

Capitolo 10: Comandi batch deprecati e loro sostituzioni

Examples

DEBUG

`DEBUG` stato utilizzato il comando `DEBUG` per creare file binari / eseguibili dal file batch. Il comando è ancora disponibile nelle versioni a 32 bit di Windows, ma è in grado di creare file binari solo con istruzioni a 16 bit, rendendolo inutilizzabile per macchine a 64 bit. Ora CERTUTIL è utilizzato a tale scopo che consente di decodificare / codificare file binari / multimediali da formati HEX o BASE64.Esempio con un file che crea file di icone:

```
@echo off

del /q /f pointer.jpg >nul 2>nul
certutil -decode "%~f0" pointer.jpg
hh.exe pointer.jpg
exit /b %errorlevel%

-----BEGIN CERTIFICATE-----
/9j/4AAQSkZJRgABAgAAZABkAAD/7AARRHVja3kAAQAEAAAAMgAA/+4ADkFkb2Jl
AGTAAAAAAf/bAIQACAYGBgYGCAYGCAwIBwgMDgoICAoOEA0NDg0NEBEMDg0NDgwR
DxITFBMSDxgYGhoYGCMiIiIjJycnJycnJycnJwEJCAgJCgkLCQkLDgsNCw4RDg4O
DhETDQ0ODQ0TGBEPDw8PERgWFxQUFBcWGhoYGBoaISEgISEnJycnJycnJycn/8AA
EQgACgAKAwEiAAIRAQMRAf/EAFsAAQEBAAAAAAAAAAAAAAAAAAAAAAAGBwEBAQAAAAAA
AAAAAAAAAAAAAAEQAAIBAwQDAAAAAAAAAAAAAAAAAAEDAgARBSExIwQSIhMRAQEBAAAA
AAAAAAAAAAAAAAAAAAARIf/aAAwDAQACEQMRAD8A13PZ5eIX3gO8ktKZfFPksvQ8r4uL
ecJmx1BMRAD8A13PZ5eIX3gO8ktKZfFPksvQ8r4uL
-----END CERTIFICATE-----
```

Ecco lo stesso con il formato esadecimale:

```
@echo off

(echo 0000    ff d8 ff e0 00 10 4a 46  49 46 00 01 02 00 00 64)   >pointer.hex
(echo 0010    00 64 00 00 ff ec 00 11  44 75 63 6b 79 00 01 00)   >>pointer.hex
(echo 0020    04 00 00 00 32 00 00 ff  ee 00 0e 41 64 6f 62 65)  >>pointer.hex
(echo 0030    00 64 c0 00 00 00 01 ff  db 00 84 00 08 06 06 06)   >>pointer.hex
(echo 0040    06 06 08 06 06 08 0c 08  07 08 0c 0e 0a 08 08 0a)   >>pointer.hex
(echo 0050    0e 10 0d 0d 0e 0d 0d 10  11 0c 0e 0d 0d 0e 0c 11)   >>pointer.hex
(echo 0060    0f 12 13 14 13 12 0f 18  18 1a 1a 18 18 23 22 22)   >>pointer.hex
(echo 0070    22 23 27 27 27 27 27 27  27 27 27 27 01 09 08 08)   >>pointer.hex
(echo 0080    09 0a 09 0b 09 09 0b 0e  0b 0d 0b 0e 11 0e 0e 0e)   >>pointer.hex
(echo 0090    0e 11 13 0d 0d 0e 0d 0d  13 18 11 0f 0f 0f 0f 11)   >>pointer.hex
(echo 00a0    18 16 17 14 14 14 17 16  1a 1a 18 18 1a 1a 21 21)   >>pointer.hex
(echo 00b0    20 21 21 27 27 27 27 27  27 27 27 27 27 ff c0 00)   >>pointer.hex
(echo 00c0    11 08 00 0a 00 0a 03 01  22 00 02 11 01 03 11 01)   >>pointer.hex
(echo 00d0    ff c4 00 5b 00 01 01 01  00 00 00 00 00 00 00 00)   >>pointer.hex
(echo 00e0    00 00 00 00 00 00 06 07  01 01 01 00 00 00 00 00)   >>pointer.hex
(echo 00f0    00 00 00 00 00 00 00 00  00 00 01 10 00 02 01 03)   >>pointer.hex
(echo 0100    04 03 00 00 00 00 00 00  00 00 00 00 01 03 02 00)   >>pointer.hex
```

```
(echo 0110    11 05 21 31 23 04 12 22  13 11 01 01 01 00 00 00)   >>pointer.hex
(echo 0120    00 00 00 00 00 00 00 00  00 00 00 11 21 ff da 00)   >>pointer.hex
(echo 0130    0c 03 01 00 02 11 03 11  00 3f 00 d7 73 d9 e5 e2)   >>pointer.hex
(echo 0140    17 de 03 bc 92 d2 99 7c  53 e4 b2 f4 3c af 8b 8b)   >>pointer.hex
(echo 0150    79 c2 66 c7 50 4c 49 b9  bc 0f a5 15 29 57 20 ff)   >>pointer.hex
(echo 0160    d9                                                  )   >>pointer.hex

certutil -decodehex "pointer.hex" pointer.jpg
hh.exe pointer.jpg
exit /b %errorlevel%
```

Come puoi vedere, esadecimale richiede file temporanei aggiuntivi e le espansioni in formato esadecimale sono più grandi

AGGIUNGERE

`APPEND` era il comando in msdos che permetteva di usare file di risorse / media come se fossero nella stessa directory. Il comando è ancora disponibile nelle versioni a 32 bit di Windows ma sembra non funzioni. In alcune fonti (compresi i microsoft ') si indica che il comando è sostituito da DPATH, ma non è completamente vero. Nonostante il messaggio di aiuto di DPATH punti al comando APPEND, il syntnax è lo stesso di `PATH` . Le directory elencate in DPATH possono essere utilizzate con il reindirizzamento dell'input o il comando type :

```
@echo off
dpath %windir%

set /p var=<win.ini
echo using dpath with input redirection:
echo %var%
echo.
echo using dpath with type command:
type win.ini
```

bitsadmin

`BITSADMIN` stato utilizzato per trasferire documenti, scaricare siti Web, scaricare file da siti Web, ecc. Questo comando è un comando deprecato e potrebbe essere rimosso nei successivi aggiornamenti di Windows. Per evitare questo problema, utilizzare il nuovo `BIT cmdlet` Powershell .

Ecco un codice di esempio che utilizza `BITSADMIN` .

```
@echo off
Bitsadmin /create /download Stackoverflow.com
rem download the website StackOverflow.com
```

Leggi Comandi batch deprecati e loro sostituzioni online: https://riptutorial.com/it/batch-file/topic/10109/comandi-batch-deprecati-e-loro-sostituzioni

Capitolo 11: Commenti nei file batch

introduzione

I commenti vengono utilizzati per mostrare le informazioni in uno script batch.

Sintassi

- REM
- & REM
- ::
- & ::
- Vai a: etichetta

```
    Comments. You can also use |>< ,etc.
```

:Etichetta

Examples

Usare REM per i commenti

```
REM This is a comment
```

- `REM` è il comando di commento ufficiale.

Utilizzo di etichette come commenti

```
::This is a label that acts as a comment
```

Il doppio carattere `::` commento mostrato sopra non è documentato come un comando di commento, ma è un caso speciale di un'etichetta che funge da commento.

Attenzione : quando le etichette vengono utilizzate come commenti all'interno di un blocco di codice tra parentesi o `for` comando, il processore di comandi si aspetta che ogni etichetta sia seguita da almeno un comando, quindi quando si esegue un salto sull'etichetta avrà qualcosa da eseguire.

La shell `cmd` proverà ad eseguire la seconda riga anche se è stata formattata come etichetta (e **questo provoca un errore**):

```
(
echo This example will fail
:: some comment
)
```

Quando si lavora all'interno di blocchi di codice con bracketing, è sicuramente più sicuro usare REM per tutte le righe di commento.

Uso delle variabili come commenti

È anche possibile utilizzare variabili come commenti. Questo può essere utile per impedire condizionalmente l'esecuzione di comandi:

```
@echo off
setlocal
if /i "%~1"=="update" (set _skip=) Else (set _skip=REM)
%_skip% copy update.dat
%_skip% echo Update applied
...
```

Quando si utilizza lo snippet di codice sopra riportato in un file batch, le righe che iniziano con `%_skip%` vengono eseguite solo se il file batch viene chiamato con `update` come parametro.

Blocca commenti

Il formato del file batch non ha una sintassi del commento del blocco, ma c'è una soluzione facile per questo.

Normalmente ogni riga di un file batch viene letta e quindi eseguita dal parser, ma è possibile utilizzare un'istruzione `goto` per saltare oltre un blocco di testo normale (che può essere usato come commento di blocco):

```
@echo off
goto :start

A multi-line comment block can go here.
It can also include special characters such as | >

:start
```

Dal momento che il parser non vede mai le linee tra l'istruzione `goto :start` e `:start` label può contenere testo arbitrario (inclusi i caratteri di controllo senza la necessità di evitarli) e il parser non genera un errore.

Commenta la riga del codice

Per commentare sulla stessa riga del codice puoi usare `&::` o `&rem` . Puoi anche usare `&&` or `||` sostituire `&` .

Esempio :

```
@echo off
echo This is a test &::This is a comment
echo This is another test &rem This is another comment
pause
```

Una curiosità : il comando `SET` consente commenti in linea *limitati* senza `&rem` :

```
set "varname=varvalue"    limited inline comment here
```

limitazioni:

- sintassi con virgolette `set "varname=varvalue"` o `set "varname="` ,
- un commento in linea **non può** contenere virgolette doppie,
- qualsiasi `cmd` caratteri velenosi `| < > &` **deve essere** correttamente scappato come `^| ^< ^> ^&` ,
- parentesi `( )` **deve essere** debitamente escapato come `^( ^)` all'interno di un blocco di codice tra parentesi.

Batch e commento ibrido WSF

```
<!-- : Comment
```

Funziona con script batch e WSF. Il tag di chiusura (`-->`), funziona solo in WSF.

Codice	Successo sia in batch che in WSF?
`<!--: Comment`	Vero
`<!--: Comment -->`	False: il tag di chiusura funziona solo per WSF
`-->`	falso

Leggi Commenti nei file batch online: https://riptutorial.com/it/batch-file/topic/3152/commenti-nei-file-batch

Capitolo 12: Creazione di file tramite batch

introduzione

Una funzione utile dei file batch è la possibilità di creare file con loro. Questa sezione mostra come creare file usando il codice batch.

Sintassi

- echo (digita qui ciò che vuoi nell'essere) >> (nomefile)
- echo (nome variabile) >> (nome file)

Osservazioni

Se esiste un file, `>` sovrascriverà il file e `>>` verrà aggiunto alla fine del file. Se un file non esiste, entrambi creeranno un nuovo file.

Inoltre, il comando `echo` aggiunge automaticamente una nuova riga dopo la stringa.

Così

```
echo 1 > num.txt
echo 1 > num.txt
echo 2 >> num.txt
```

creerà il seguente file:

```
1
2
```

Non questo:

```
1 1 2
```

o

```
1 2
```

Inoltre, non è possibile modificare una singola riga in un file di testo. Devi leggere l'intero file, modificarlo nel tuo codice e poi scrivere di nuovo sull'intero file.

Examples

reindirizzamento

Formato:

```
[command] [> | >>] [filename]
```

`>` *salva* l'output di [comando] in [nomefile].

`>>` *aggiunge* l'output di [comando] in [nomefile].

Esempi:

1. `echo Hello World > myfile.txt` salva "Hello World" in myfile.txt
2. `echo your name is %name% >> myfile.txt` aggiunge "il tuo nome è xxxx" in myfile.txt
3. `dir C:\ > directory.txt` salva la directory di C: \ in directory.txt

Echo per creare file

Modi per creare un file con il comando echo:

```
ECHO. > example.bat (creates an empty file called "example.bat")

ECHO message > example.bat (creates example.bat containing "message")
ECHO message >> example.bat (adds "message" to a new line in example.bat)
(ECHO message) >> example.bat (same as above, just another way to write it)
```

Se si desidera creare un file tramite il comando `ECHO` , in una directory specifica sul computer, si potrebbe incorrere in un problema.

```
ECHO Hello how are you? > C:\Program Files\example.bat

(This will NOT make a file in the folder "Program Files", and might show an error message)
```

Ma allora come lo facciamo? Beh, in realtà è estremamente semplice ... Quando digiti un percorso o un nome di file che ha uno spazio incluso nel suo nome, ricorda di usare "virgolette"

```
ECHO Hello how are you? > "C:\Program Files\example.bat"
(This will create "example.bat" in the folder "Program Files")
```

Ma cosa succede se si desidera creare un file che emette un nuovo file?

```
ECHO message > file1.bat > example.bat

(example.bat is NOT going to contain "message > file1.bat")
example.bat will just contain "message"... nothing else
```

Allora come lo facciamo? Bene per questo usiamo il simbolo `^` .

```
ECHO message ^> file1.bat > example.bat
```

```
(example.bat is going to contain "message > file1.bat")
```

Lo stesso vale per altre cose in batch

Il prossimo passo richiede di avere una certa conoscenza di variabili e dichiarazioni:

clicca qui per conoscere le variabili | clicca qui per sapere se e else dichiarazioni

variabili:

```
SET example="text"
ECHO %example% > file.bat
(This will output "text" to file.bat)
```

se non vogliamo che produca "testo" ma semplicemente% esempio% quindi scrivi:

```
ECHO ^%example^% > file.bat
(This will output "%example%" to file.bat)
```

Istruzioni IF / ELSE:

```
ELSE statements are written with "pipes" ||

IF ^%example^%=="Hello" ECHO True || ECHO False > file.bat

(example.bat is going to contain "if %example%=="Hello" echo True")
[it ignores everything after the ELSE statement]
```

per produrre l'intera linea, facciamo lo stesso di prima.

```
IF ^%example^%=="Hello" ECHO True ^|^| ECHO False > file.bat

This will output:
IF %example%=="Hello" ECHO True || ECHO False
```

Se la variabile è uguale a "Hello", allora dirà "True", ELSE dirà "False"

Salvataggio dell'output di molti comandi

Utilizzo di molti comandi `ECHO` per creare un file batch:

```
(
  echo echo hi, this is the date today
  echo date /T
  echo echo created on %DATE%
  echo pause >nul
)>hi.bat
```

L'interprete dei comandi tratta l'intera sezione tra parentesi come un singolo comando, quindi salva tutto l'output su `hi.bat` .

`hi.bat`

ora contiene:

```
echo hi, this is the date today
date /T
echo created on [date created]
pause >nul
```

Leggi Creazione di file tramite batch online: https://riptutorial.com/it/batch-file/topic/7129/creazione-di-file-tramite-batch

Capitolo 13: Differenze tra Batch (Windows) e Terminal (Linux)

introduzione

Batch e bash sono abbastanza diversi. I flag batch sono indicati con un `/` , mentre i flag bash usano un `-` . La capitalizzazione conta in bash, ma (quasi) per nulla in batch. I nomi delle variabili batch possono contenere spazi, i nomi delle variabili bash non possono. In definitiva, entrambi sono modi di manipolare e interagire con la riga di comando. Non sorprendentemente, vi è una quantità abbastanza grande di sovrapposizione tra le capacità dei due sistemi.

Osservazioni

- `bitsadmin` è deprecato a favore del cmdlet BITS di PowerShell ma funziona ancora da Windows 10 versione 1607
- `certutil` separa le coppie di cifre esadecimali con uno spazio, quindi `md5sum` restituirà un valore di esempio di `d41d8cd98f00b204e9800998ecf8427e` , mentre `certutil` visualizza lo stesso valore di `d4 1d 8c d9 8f 00 b2 04 e9 80 09 98 ec f8 42 7e`
- Per eseguire il `cd` su un'altra unità (ad esempio, da C: a D :), è necessario utilizzare l'indicatore `/d`
- `del` non può eliminare cartelle, usa invece `rm`
- `grep` è molto più potente di `find` e `findstr` , non è quasi corretto confrontarli; `find` non ha capacità regex e `findstr` ha capacità regex estremamente limitate (`[az]{2}` non è una sintassi valida, ma `[az][az]` è)
- `for` cicli sul prompt dei comandi di Windows è possibile utilizzare solo nomi di variabili a singolo carattere; questa è l'unica volta in cui i nomi delle variabili batch sono sensibili al maiuscolo / minuscolo
- `for` cicli sul prompt dei comandi utilizzare anche la forma variabile `%A` anziché `%A%` - `for` loop negli script batch utilizzare la forma variabile `%%A`
- `md` crea automaticamente tutte le directory madri necessarie, mentre `mkdir` bisogno del flag `-p` per farlo
- `rem` non può essere usato come un carattere di commento in linea a meno che non sia preceduto da un `&`
- `::` non può essere usato come commento in linea e non deve essere usato all'interno di un blocco di codice (insieme di parentesi)

- Si noti che alcuni dei comandi di Windows come `ping` utilizza ancora `-` come bandiere

Examples

Comandi batch e loro equivalenti Bash

pa ta	bash	Descrizione
`command /?`	`man command`	Mostra l'aiuto per il *comando*
`bitsadmin`	`wget` o `curl`	Scarica un file remoto
`certutil -hashfile file_name MD5`	`md5sum file_name`	Ottiene il checksum MD5 di *nome_file*
`cd`	`pwd`	Visualizza la directory corrente
`cd directory`	`cd directory`	Cambia la directory corrente in quella specificata
`cls`	`clear`	Cancella lo schermo
`copy`	`cp`	Copia un file o file da un percorso di origine a un percorso di destinazione
`date`	`date`	Visualizza la data o la imposta in base all'input dell'utente
`del`	`rm`	Elimina un file o file
`dir`	`ls`	visualizza un elenco di file e directory nella directory corrente
`echo`	`echo`	Visualizza il testo sullo schermo
`exit`	`return`	Esce da uno script o subroutine
`exit`	`logout`	Chiude il prompt dei comandi o il terminale
`fc`	`diff`	Confronta i contenuti di due file
`find "string" file_name`	`grep "string" file_name`	Cerca *file_name* per *stringa*
`findstr "string" file_name`	`grep "string" file_name`	Cerca *file_name* per *stringa*
`for /F %A in (fileset*) do something`	`for item in fileset*; do; something; done`	Fai qualcosa per ogni file in un set di file
`for /F %A in ('command') do something`	`` `command` ``	Restituisce l'output di un comando
`for /L %A in (first,increment,last) do something`	`for item in `seq first increment last`; do; something; done`	Inizia *all'inizio* e conta per *incrementi* fino a raggiungere l' *ultimo*

pa...ta	bash	Descrizione
`forfiles`	`find`	Cerca i file che corrispondono a un determinato criterio
`if "%variable%"=="value" (`	`if [ "variable"="value" ]; then`	Confronta due valori
`ipconfig`	`ifconfig`	Visualizza le informazioni IP
`md`	`mkdir`	Crea nuove cartelle
`mklink`	`ln -s`	Crea un collegamento simbolico
`more`	`more`	Visualizza una schermata di output alla volta
`move`	`mv`	Sposta un file o file da un percorso di origine a un percorso di destinazione
`pause`	`read -p "Press any key to continue"`	Mette in pausa l'esecuzione dello script fino a quando l'utente preme un pulsante
`popd`	`popd`	Rimuove la voce superiore dallo stack di directory e passa alla nuova directory superiore
`pushd`	`pushd`	Aggiunge la directory corrente allo stack di directory e passa alla nuova directory superiore
`ren`	`mv`	Rinomina i file
`rem` o `::`	`#`	Commenti una riga di codice
`rd`	`rmdir`	Rimuove le directory vuote
`rd /s`	`rm -rf`	Rimuove le directory indipendentemente dal fatto che siano o meno vuote
`set variable=value`	`variable=value`	Imposta il valore della *variabile* su *valore*
`set /a variable=equation`	`variable=$((equation))`	Esegue matematica (batch può usare solo numeri interi a 32 bit)
`set /p variable=promptstring`	`read -p "promptstring" variable`	Ottiene l'input dell'utente e lo memorizza in *variabile*

partita	bash	Descrizione
`shift`	`shift`	Sposta gli argomenti di 1 (o n se fornito)
`sort`	`sort`	Ordina l'output in ordine alfabetico per riga
`tasklist`	`ps`	Mostra un elenco di processi in esecuzione
`taskkill /PID processid`	`kill processid`	Uccide il processo con *processid* PID
`time /t`	`date`	Visualizza l'ora corrente
`type`	`cat`	Visualizza il contenuto di un file
`where`	`which`	Cerca la directory corrente e il PERCORSO per un file o comando
`whoami`	`id`	Visualizza il nome e il gruppo dell'utente corrente

Variabili batch e il loro equivalente Bash

partita	bash	Descrizione
`%variable%`	`$variable`	Una variabile regolare
`!variable!`	`$variable`	Una variabile all'interno di un blocco di codice quando è attiva l' `setlocal enabledelayedexpansion`
`%errorlevel%` o `ERRORLEVEL`	`$?`	Restituisce lo stato del comando precedente: 0 se ha successo, 1 (o qualcos'altro) se non lo è
`%1` , `%2` , `%3` , ecc.	`$1` , `$2` , `$3` , ecc.	I parametri dati a uno script
`%*`	`$*`	Tutti i parametri dati a uno script

Leggi Differenze tra Batch (Windows) e Terminal (Linux) online: https://riptutorial.com/it/batch-file/topic/8362/differenze-tra-batch--windows--e-terminal--linux-

Capitolo 14: Eco

introduzione

`echo` può essere usato per controllare e produrre output.

Sintassi

- ECHO [ON | OFF]
- Messaggio ECHO
- ECHO (messaggio
- ECO(

Parametri

Parametro	Dettagli
ON | OFF	Può essere `ON` o `OFF` (senza distinzione tra maiuscole e minuscole)
Messaggio	Qualsiasi stringa (tranne `ON` o `OFF` se usato senza `(`)

Osservazioni

- `echo.` mostrerà anche una stringa vuota. Tuttavia, questo è più lento di `echo(` come `echo.` cercherà un file chiamato "echo". Solo se questo file non esiste il comando funzionerà, ma questo controllo lo rende più lento.
- `echo:` si comporterà proprio come `echo(` , a meno che il `message` non assomigli ad un percorso file, ad esempio `echo:foo\..\test.bat` . In questo caso, l'interprete vedrà `echo:foo` come nome di una cartella, strip `echo:foo\..\` (perché appare solo per entrare nella directory `echo:foo` quindi lasciarlo di nuovo) quindi eseguire `test.bat` , che non è il comportamento desiderato.

Examples

Visualizzazione dei messaggi

Per visualizzare "Some Text", usa il comando:

```
echo Some Text
```

Questo produrrà la stringa `Some Text` seguita da una nuova riga.

Per visualizzare le stringhe `On` e `Off` (senza distinzione tra maiuscole e minuscole) o la stringa vuota, utilizzare un `(` anziché uno spazio bianco:

```
echo(ON
echo(
echo(off
```

Questo produrrà:

```
ON

off
```

È anche comune usare `echo.` per generare una riga vuota, ma per favore vedi le osservazioni sul motivo per cui questa non è la migliore idea.

Per visualizzare il testo senza includere un CR / LF, utilizzare il seguente comando:

```
<nul set/p=Some Text
```

Questo comando tenterà di impostare la variabile chiamata stringa vuota sull'input dell'utente dopo un prompt. Il file `nul` viene reindirizzato al comando con `<nul` , quindi il comando si arrenderà non appena tenta di leggere da esso e verrà lasciata solo la stringa del prompt. Perché l'utente non ha mai digitato una nuova riga, non c'è alcun avanzamento.

Impostazione dell'eco

L'impostazione dell'eco determina se l'eco del comando è attivato o disattivato. Ecco come appare un programma di esempio con il comando echoing **on** (predefinito):

```
C:\Windows\System32>echo Hello, World!
Hello, World!

C:\Windows\System32>where explorer
C:\Windows\System32\explorer.exe

C:\Windows\System32>exit
```

Questo è ciò che appare con l'eco **spento** :

```
Hello, World!
C:\Windows\System32\explorer.exe
```

Ottenimento e impostazione

Per ottenere (visualizzare) l'impostazione dell'eco, usa `echo` senza parametri:

```
> echo
ECHO is on.
```

Per impostare l'eco, usa `echo` con `on` o `off` :

```
> echo off

> echo
ECHO is off.

> echo on

> echo
ECHO is on.
```

Si noti che con questi esempi, il prompt è stato rappresentato da `>` . Quando si modifica l'impostazione dell'eco, il prompt apparirà e scomparirà, ma questo rende gli esempi non chiari.

Notare che è possibile impedire che un comando venga echeggiato anche quando è attivo echo, posizionando un carattere `@` prima del comando.

Echo emette tutto letteralmente

Le virgolette saranno emesse così com'è:

```
echo "Some Text"
```

```
"Some Text"
```

I token di commento vengono ignorati:

```
echo Hello World REM this is not a comment because it is being echoed!
```

```
Hello World REM this is not a comment because it is being echoed!
```

Tuttavia, `echo` continuerà a generare variabili - vedi Uso delle variabili - e i caratteri speciali avranno ancora effetto:

```
echo hello && echo world
```

```
hello
world
```

Uscita dell'eco su file

Modi per creare un file con il comando echo:

```
echo. > example.bat (creates an empty file called "example.bat")

echo message > example.bat (creates example.bat containing "message")
echo message >> example.bat (adds "message" to a new line in example.bat)
(echo message) >> example.bat (same as above, just another way to write it)
```

Uscita sul percorso

Un piccolo problema che potresti incontrare quando fai questo:

```
echo Hello how are you? > C:\Users\Ben Tearzz\Desktop\example.bat

(This will NOT make a file on the Desktop, and might show an error message)
```

Ma allora come lo facciamo? Beh, in realtà è estremamente semplice ... Quando digiti un percorso o un nome di file che ha uno spazio incluso nel suo nome, ricorda di usare "virgolette"

```
echo Hello how are you? > "C:\Users\Ben Tearzz\Desktop\example.bat"
(This will make a file on MY Desktop)
```

Ma cosa succede se si desidera creare un file che emette un nuovo file?

```
echo message > file1.bat > example.bat

(This is NOT going to output:
"message > file1.bat" to example.bat
```

Allora come lo facciamo?

```
echo message ^> file1.bat > example.bat

(This will output:
"message > file1.bat" to example.bat
```

Lo stesso vale per altre cose in batch

Se non hai imparato quali sono le variabili e le affermazioni, probabilmente non capirai quanto segue: clicca qui per conoscere le variabili | clicca qui per conoscere le dichiarazioni "if"

variabili:

```
set example="text"
echo %example% > file.bat
(This will output "text" to file.bat)
```

se non vogliamo che produca "testo" ma semplicemente% esempio% quindi scrivi:

```
echo ^%example^% > file.bat
(This will output "%example%" to file.bat)
```

altre dichiarazioni:

```
else = ||
if ^%example^%=="Hello" echo True || echo False > file.bat

(This will output:
if %example%=="Hello" echo True
```

per produrre l'intera riga scriviamo:

```
if ^%example^%=="Hello" echo True ^|^| echo False > file.bat

This will output:
if %example%=="Hello" echo True || echo False
```

Se la variabile è uguale a "Hello", allora dirà "True", altrimenti dirà "False"

@ Elimina

`@echo off` impedisce la visualizzazione del prompt e del contenuto del file batch, in modo che solo l'output sia visibile. Il `@` rende anche nascosto l'output del comando `echo off` .

Girando l'eco sulle parentesi interne

Nell'esempio seguente, l' `echo on` avrà effetto dopo il raggiungimento del fine del contesto delle parentesi:

```
@echo off
(
    echo on
    echo ##
)
echo $$
```

Per "attivare" l'eco in un contesto di parentesi (inclusi i comandi `FOR` e `IF`), è possibile utilizzare FOR / f macro :

```
@echo off
setlocal

:: echo on macro should followed by the command you want to execute with echo turned on
set "echo_on=echo on&for %%. in (.) do"

(
  %echo_on% echo ###
)
```

Leggi Eco online: https://riptutorial.com/it/batch-file/topic/3981/eco

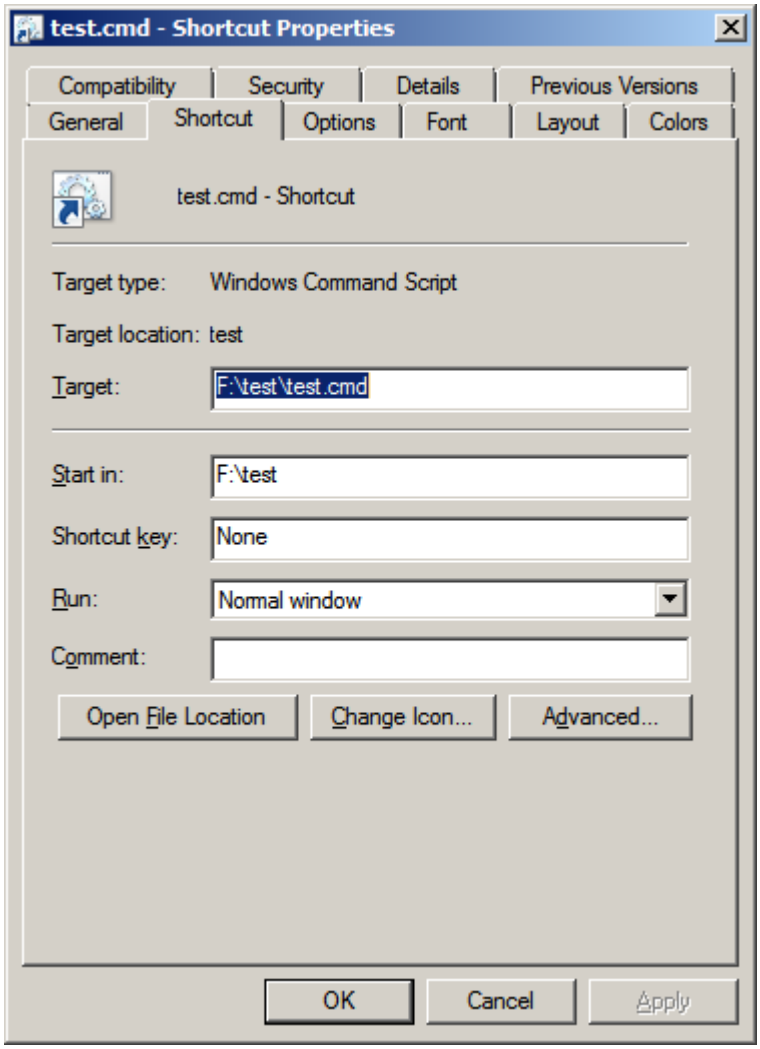
Capitolo 15: Elevati privilegi nei file batch

Examples

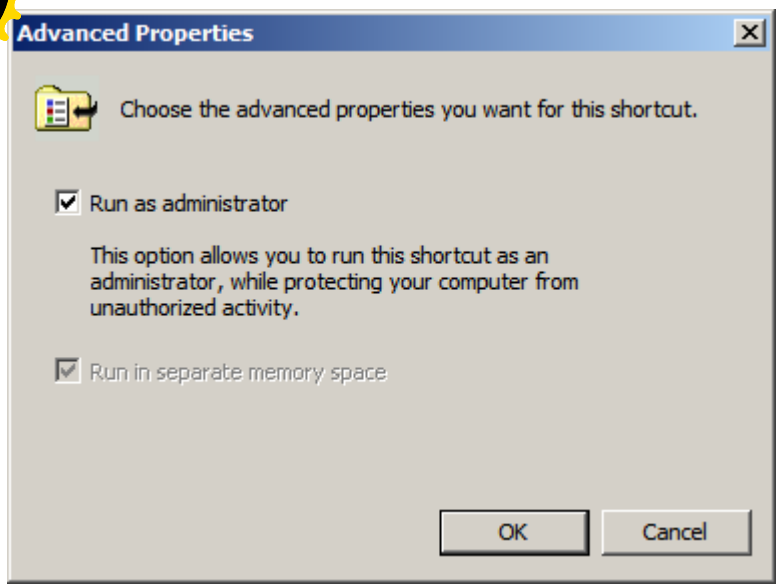
Richiesta di elevare i privilegi in una scorciatoia

Molte attività richiedono privilegi elevati. È possibile elevare i privilegi dell'utente a livello di amministratore per il runtime batch utilizzando una scorciatoia:

1. Premi `alt` + e il file batch in una cartella selezionata per creare un collegamento.
2. Fare clic con il tasto destro e selezionare "Proprietà".
3. Seleziona la scheda "Collegamento".
4. Fai clic su "Avanzate".



5. Abilita "Esegui come amministratore".



6. Fai clic su "OK" due volte.

Richiesta di privilegi elevati in fase di runtime

Il seguente file batch visualizzerà un prompt UAC che consente di accettare privilegi di amministratore elevati per la sessione batch. Aggiungi il tuo codice di attività a `:usercode` sezione del codice `:usercode` del batch, in modo che vengano eseguiti con privilegi elevati.

```
@echo off
setlocal EnableDelayedExpansion
:: test and acquire admin rights
cd /d %~dp0 & echo/
if not "%1"=="UAC" (
    >nul 2>&1 net file && echo Got admin rights || (echo No admin rights & ^
MSHTA "javascript: var shell = new ActiveXObject('shell.application');
shell.ShellExecute("%~snx0", 'UAC', '', 'runas', 1);close();"))
:: re-test admin rights
echo/ & >nul 2>&1 net file && (echo Got admin rights & echo/) || (echo No admin rights.
Exiting... & goto :end)

:usercode
:: add your code here
echo Performing admin tasks
echo Hello >C:\test.txt

:end
timeout /t 5 >nul
exit /b
```

Richiesta di privilegi elevati di runtime senza prompt UAC

Come esempio precedente, questa elevazione richiesta di script, se necessario. Chiediamo all'utente le credenziali evitando il prompt UAC.

```
@echo off

cls & set "user=" & set "pass="
```

```
:: check if we have administrative access
:: ---------------------------------------------------------------------------
whoami /groups | find "S-1-5-32-544">NUL 2>&1 && goto:gotAdmin


echo/
echo/ Testing administrative privileges
echo/
echo/ Please enter administrative credentials
echo/

:: get user
:: ---------------------------------------------------------------------------
set/P user="*     User:: "
if "%user%" EQU "" exit/B 1
:: get password
:: ---------------------------------------------------------------------------
<NUL set/p=* password& call:getPass pass || exit/B 1
if "%pass%" EQU "" exit/B 1

:: check if credential has administrative privileges
:: this call can be removed
:: ---------------------------------------------------------------------------
call:askIsAdmin "%user%", "%pass%" || goto:notAdmin

:: run elevated without UAC prompt
:: with the previous call enabled, we are sure credentials are right
:: without it, this will fail if credentials are invalid
:: ---------------------------------------------------------------------------
call:elevateScript "%user%", "%pass%"

exit/B 0

:gotAdmin
echo(
echo( You have administrative rights.
echo(
timeout /T 7
exit/B 0

:notAdmin
echo(
echo( Invalid credential or non administrative account privileges
echo(
timeout /T 7
exit/B 1

:: invoke powershell to get the password
:: ---------------------------------------------------------------------------
:getPass
SetLocal
set "psCmd=powershell -Command "$pwd = read-host ':' -AsSecureString;
$BSTR=[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($pwd);
[System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)""
for /F "usebackq delims=" %%# in (`%psCmd%`) do set "pwd=%%#"
if "%pwd%" EQU "" EndLocal & exit/B 1
EndLocal & set "%1=%pwd%"
doskey /listsize=0 >NUL 2>&1 & doskey /listsize=50 >NUL 2>&1        & rem empty keyboard
buffer
exit/B 0
```

```
:: check if credential has administrative privileges
:: -------------------------------------------------------------------------
:askIsAdmin
SetLocal & set "u=%~1" & set "p=%~2" & set/A ret=1
set "psCmd=powershell -Command "$p='%p%'^|convertto-securestring -asplaintext -force;$c=new-
object -typename system.management.automation.pscredential('%u%',$p^);start-process
'powershell' '-Command "write-host ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent(^)^).IsInRole([Security.Principal.WindowsBuiltInRole]
-credential $c -passthru -wait;""
for /F "usebackq delims=" %%# in (`%psCmd%`) do @set "result=%%#"
echo %result% | find /I "true">NUL 2>&1 && set/A ret=0
EndLocal & exit/B %ret%
exit/B 1

:: run elevated without UAC prompt
:: -------------------------------------------------------------------------
:elevateScript
SetLocal & set "u=%~1" & set "p=%~2"
set "_vbs_file_=%TEMP%\runadmin.vbs"
echo set oWS ^= CreateObject^("wScript.Shell"^)>"%_vbs_file_%"
echo strcmd="C:\Windows\system32\runas.exe /user:%COMPUTERNAME%\%u% " +
"""%~dpnx0""">>"%_vbs_file_%"
echo oWS.run strcmd, 2, false>>"%_vbs_file_%"
echo wScript.Sleep 100>>%_vbs_file_%
echo oWS.SendKeys "%p%{ENTER}">>%_vbs_file_%
if exist "%TEMP%\runadmin.vbs" (set "_spawn_=%TEMP%\runadmin.vbs") else (set
"_spawn_=runadmin.vbs")
ping 1.1.1.1 -n 1 -w 50 >NUL
start /B /WAIT cmd /C "cls & "%_spawn_%" & del /F /Q "%_spawn_%" 2>NUL"
EndLocal & set "pass="
exit/B 0
```

Leggi Elevati privilegi nei file batch online: https://riptutorial.com/it/batch-file/topic/4898/elevati-privilegi-nei-file-batch

Capitolo 16: Escaping caratteri speciali

introduzione

In tutte le versioni `cmd.exe` e `DOS` , alcuni caratteri sono riservati per un uso specifico (ad esempio il reindirizzamento dei comandi). Questo argomento parlerà di come utilizzare i caratteri speciali senza problemi.

Examples

Fuga usando il segno di omissione (^)

La maggior parte dei caratteri speciali può essere sfuggita utilizzando il segno di omissione (`^`). Dai un'occhiata al seguente esempio.

```
echo > Hi
echo ^> Hi
```

Questo primo comando non verrà emesso `> Hi` perché `>` è un carattere speciale, che significa reindirizzare l'output a un file. In questo caso, il file è denominato "Ciao"

Comunque nel secondo comando, `> Hi` sarebbe emesso senza alcun problema perché il segno di omissione (`^`) dice al comando `>` di smettere di funzionare come comando "redireziona l'output su file", ora `>` è solo un carattere normale.

Ecco una lista di caratteri speciali che possono essere sfuggiti (presi e modificati dalla pagina di Rob van der Woude)

Personaggio	Risultato sfuggito	Osservazioni
^	^^	
&	^ &	
<	^ <	
>	^>	
|	^ |	
\	^ \	
!	^^!	Richiesto solo quando DelayedExpansion è attivo

Sfuggire al cursore

I carnet possono essere impilati fino alla fuga di altri caret, prendere in considerazione il seguente esempio.

Ingresso	Produzione
^ &	&
^ **^** ^ &	^ &
^ **^** ^ **^** ^ &	^^ &

Nota: i caratteri in grassetto sono sfuggiti.

Problema di sicurezza

Un po 'fuori tema qui, ma questo è molto importante! Una fuga del cursore indesiderata alla fine del file potrebbe causare una perdita di memoria!

```
any-invalid-command-you-like-here ^
```

Questo comando potrebbe perdere tutta la memoria, **rendendo il sistema completamente inutilizzabile** ! Vedi qui per maggiori informazioni.

TROVA e FINDSTR Caratteri speciali

In `find` e `findstr` , ci sono alcuni caratteri speciali che richiedono una certa cautela.

TROVA

C'è solo un personaggio che ha bisogno di scappare - `"` preventivo `"` Per evitarlo, aggiungi semplicemente un'altra citazione accanto ad esso, quindi `"` diventa `""` . Abbastanza semplice.

FINDSTR

`Findstr` viene fornito con un sacco di personaggi da cui fuggire, quindi per favore sii molto cauto. Usando `\` , possiamo sfuggire a caratteri speciali. Ecco una lista di personaggi speciali da sfuggire

Personaggio	Risultato sfuggito
\	\\
[	\ [
]	\]
"	\"
.	\.
*	\ *
?	\?

Caratteri speciali FOR / F

FOR / F

In una frase `FOR /F` , alcuni caratteri devono essere salvati, qui una lista (presa e modificata dalla pagina di Rob van der Woude)

Personaggio	Risultato sfuggito	Osservazioni
'	^'	`usebackq` solo nelle parentesi `FOR /F` , a meno che non sia specificato `usebackq` .
`	^ `	`usebackq` solo nelle parentesi `FOR /F` , quando viene specificato `usebackq`
,	^,	┐
;	^;	┃
=	^ =	┣ Deve essere sfuggito nelle parentesi `FOR /F` , anche se è doppia citazione
(	^ (	┃
)	^)	┘

Caratteri speciali extra

Ecco una lista di altri caratteri speciali, che richiedono (s) / potrebbe essere necessario fuggire, ma non menzionati sopra.

Personaggio	Risultato sfuggito	Osservazioni
%	%%	
[LF]	^ [LF]	Questo trucco è interpretato da Mark Stang nel gruppo di notizie `alt.msdos.batch` .

Fuggire attraverso il gasdotto

Quando c'è un'espressione con una pipe, il `cmd` avvia due thread su entrambi i lati della pipe e l'espressione viene analizzata due volte (per ciascun lato della pipe) in modo che i segni di omissione debbano essere raddoppiati.

Sul lato sinistro:

```
echo ^^^&|more
```

Dal lato giusto:

```
break|echo ^^^&
```

Leggi Escaping caratteri speciali online: https://riptutorial.com/it/batch-file/topic/10693/escaping-caratteri-speciali

Capitolo 17: File batch e ibridi PowerShell

Examples

Esegui PowerShell con i file temporanei

Questo è stato menzionato più e più volte in altri argomenti ibridi . La vecchia scuola, ma il metodo semplice per eseguire Powershell è di:

- `echo` lo script PowerShell in uno script temporaneo
- Esegui lo script temporaneo
- Rimuovere facoltativamente lo script temporaneo

Questo è uno script di esempio.

```
@echo off
echo powershell-command>Temp.ps1
echo another line>>Temp.ps1
    rem echo the script into a temporary file

powershell -File Temp.ps1
    rem execute the temporary script

del Temp.ps1
    rem Optionally remove the temporary script
```

Il metodo sopra richiede tonnellate di istruzioni `echo` se è richiesto uno script lungo, ecco un metodo migliore suggerito da @Aacini

```
@echo off
setlocal

    rem Get the number of the "<resource>" line
for /F "delims=:" %%a in ('findstr /N "<resource>" "%~F0"') do set "start=%%a"

    rem Skip such number of lines and show the rest of this file
(for /F "usebackq skip=%start% delims=" %%a in ("%~F0") do echo %%a) > Temp.ps1

powershell -File Temp.ps1
del /f /s /q Temp.ps1

goto :EOF

<resource>
PS
Powershell script
```

Utilizzare il comando POWERSHELL per eseguire il comando PowerShell a una sola riga

Usando il comando `POWERSHELL` , possiamo eseguire un comando a 1 riga direttamente da uno script batch, senza alcun file temporaneo.

Ecco la sintassi.

```
powershell.exe -Command <yourPowershellCommandHere>
```

Potresti anche voler includere altri flag, come `-Nologo` per migliorare il risultato effettivo.

Powershell / ibrido batch senza file temporanei

Questo è l'approccio proposto dall'utente rojo di stackoverflow che può anche gestire gli argomenti della riga di comando:

```
<# : batch portion
@echo off & setlocal

(for %%I in ("%~f0";%*) do @echo(%%~I) | ^
powershell -noprofile "$argv = $input | ?{$_}; iex (${%~f0} | out-string)"

goto :EOF
: end batch / begin powershell #>

"Result:"
$argv | %{ "`$argv[{0}]: $_" -f $i++ }
```

chiamato così:

    psbatch.bat arg1 "Questo è arg2" arg3

produrrà:

```
Result:
$argv[0]: C:\Users\rojo\Desktop\test.bat
$argv[1]: arg1
$argv[2]: This is arg2
$argv[3]: arg3
```

Leggi File batch e ibridi PowerShell online: https://riptutorial.com/it/batch-file/topic/10711/file-batch-e-ibridi-powershell

Capitolo 18: funzioni

Osservazioni

Puoi aggiungere variabili iniziali alla funzione aggiungendo `<parameter>` alla sua etichetta. È possibile accedere a queste variabili iniziali con `%n` dove n è il numero della variabile iniziale (`%1` per il primo, `%2` per il `%n` metodo `%n` funziona con% 1 -% 9. Per il parametro 10 - 255, è necessario usare il comando Shift).
Per esempio:

```
:function <var1> <var2>
```

Dopo aver usato la `call :function param1 param2` , `param1` possibile accedere con `%1` e `param2` con `%2` .
Nota: il `<parameter>` non è strettamente necessario, ma aiuta con la leggibilità.

Un trucchetto che è utile quando molte variabili volano è usare `setlocal` e `endlocal` in tandem con `%n` . `setlocal` e `endlocal` rendono essenzialmente la funzione la propria istanza separata del prompt dei comandi, mentre le variabili impostate rimangono solo mentre sono nel frame.

Se stai usando `setlocal` e `endlocal` , e stai restituendo valori globali, usa questo.

```
endlocal & set var=variable
```

Questo imposta il valore globale `var` su `variable` . Puoi concatenarli insieme per più variabili.

```
endlocal & set var=variable & set var2=variable number 2
```

Questo imposta la variabile globale `var` in `variable` e il valore globale `var2` in `variable number 2` .
Poiché anche il codice nei blocchi di codice viene eseguito simultaneamente, puoi farlo anche tu.

```
if "%var%"=="" (
    endlocal
    set %~2=10
)
```

Ma non **puoi** farlo.

```
if "%var%"=="" (
    set %~2=10
    endlocal
)
```

Examples

Funzione semplice

```
call :FunctionX
rem More code...

:FunctionX
rem Some code here.
goto :eof
```

Questa è una funzione molto semplice. Le funzioni sono comandi in-programma che eseguono più comandi alla volta. Le funzioni vengono create creando un'etichetta e inserendo il codice e, una volta terminata, si aggiunge `goto :eof` o `exit /b <ErrorlevelYou'dLike>` che ritorna a dove è stato richiamato. Le funzioni sono invocate con `call :functionname adparams` .

Funzione con parametri

```
call :tohex 14 result
rem More code...

:tohex <innum> <outvar>
set dec=%1
set outvar=%~2
rem %n and %~n are functionally identical, but %~n is slightly safer.
goto :eof
```

Questo prende i parametri aggiuntivi dalla `call` come se la funzione fosse un file batch separato. Nota: il `<parameter>` non è necessario, ma aiuta con la leggibilità.

Funzione Utilizzo di setlocal e endlocal

```
set var1=123456789
set var2=abcdef
call :specialvars
echo %var1%, %var2%
rem More code...

:specialvars
setlocal
set var1=987654321
set var2=fedcba
endlocal
goto :eof
```

Quando all'interno della sezione `setlocal` , sezione `endlocal` , le variabili sono separate dalle variabili del chiamante, quindi perché% var1% e% var2% non sono stati modificati.

Combinandoli tutti

```
set importantvar=importantstuff
call :stuff 123 var1
rem More code...

:stuff <arg1> <arg2>
setlocal
set importantvar=%~1
```

```
e Writing some stuff into %~2!
endlocal
set %~2=some stuff
setlocal
set importantvar=junk
endlocal
goto :eof
```

Questo utilizza la funzione di base, `setlocal` e `endlocal` e gli argomenti per creare una strana piccola funzione.

Funzioni anonime nei file batch

La tecnica delle funzioni anonime utilizza il fatto che il comando `CALL` utilizza internamente `GOTO` quando viene chiamata la subroutine e abusa della stampa del messaggio di aiuto con doppia espansione variabile :

```
@echo off
setlocal
set "anonymous=/?"

call :%%anonymous%% a b c 3>&1 >nul

if "%0" == ":%anonymous%" (
  echo(
  echo Anonymous call:
  echo %%1=%1 %%2=%2 %%3=%3
  exit /b 0
)>&3
```

È possibile chiamare una funzione anonima solo se è definita dopo la `CALL` (o dopo aver terminato il contesto delle parentesi se la `CALL` è eseguita tra parentesi). Non può essere chiamato da uno script esterno , ma è più lento della normale chiamata di funzione.

Chiamare le funzioni da un altro file batch

Consente di avere il seguente file chiamato **library.cmd** :

```
@echo off

echo -/-/- Batch Functions Library -/-/-

:function1
    echo argument1 - %1
    goto :eof
```

Per eseguire solo **: function1** senza il codice del resto del file dovresti inserire un'etichetta **: function1** nel caller bat e usarla in questo modo:

```
@echo off

call :function1 ###
exit /b %errorlevel%
```

```
:function1
    library.bat %*
```

l'output sarà (il codice al di fuori della funzione in `library.cmd` non viene eseguito):

    argomento1 - ###

Per maggiori informazioni controlla questo .

Leggi funzioni online: https://riptutorial.com/it/batch-file/topic/7646/funzioni

Capitolo 19: Gestione file in file batch

introduzione

In questo argomento imparerai come creare, modificare, copiare, spostare ed eliminare i file in batch.

Examples

Creazione di un file in batch

Potrebbero esserci più motivi per cui si desidera creare un file di testo in batch. Ma qualunque sia la ragione, questo è il modo in cui lo fai.

Se vuoi sovrascrivere un file di testo esistente usa `>` . Esempio:

```
@echo off
echo info to save > text.txt
```

Ma se vuoi aggiungere del testo a un file di testo già esistente, usa `>>` . Esempio:

```
@echo off
echo info to save >> text.txt
```

Se è necessario salvare più righe di testo in un file, utilizzare `()>text.txt` Esempio:

```
@echo off
(echo username
 echo password)>text.txt
```

Come copiare i file in batch

Potresti voler copiare i file da un posto all'altro. In questo esempio ti insegneremo.

Puoi usare il comando `xcopy` . La sintassi è `xcopy c:\From C:\To`

Esempio:

```
@echo off
xcopy C:\Folder\text.txt C:\User\Username\Desktop
```

Ci sono anche degli switch che puoi usare. Se si desidera visualizzarli, aprire il prompt dei comandi dal `Start Menu -> Accessories -> Command Prompt` e digitare `xcopy /?`

Spostare i file

Utilizzando il comando move, puoi spostare i file:

```
@echo off
cd C:\Foo\Bat\Baz
move /Y Meow.bat "Meow Folder" >nul
```

`Meow.bat` indica il file da spostare. La "cartella Meow" sposta `Meow.bat` nella `Meow Folder` . `/Y` dice di non richiedere conferma. Sostituendolo con `/-Y` , il prompt del file batch richiede conferma. Il `>nul` nasconde l'output del comando. Se non avesse `>nul` , produrrebbe:

```
    1 File Moved
```

Eliminazione di file

Utilizzando il comando `DEL` (alias per `ERASE`), è possibile rimuovere i file.

```
@echo off
del foo.ext
```

Questo comando cancellerà `foo.ext` dalla directory corrente. Si può anche specificare il percorso e il file, come ad esempio:

```
del C:\Foo\Bar\Baz.ext
```

Ma è sempre l'ideale per mettere le virgolette (`"`) attorno ai percorsi, vedere qui per il motivo.

Sono disponibili alcuni flag per `DEL` .

Bandiera	Funzione
`/P`	Richiede all'utente prima di eliminare i file
`/F`	Rimuovere con forza i file di sola lettura
`/S`	Rimuovi file (s) in sottodirectory
`/Q`	Previene il prompt dell'utente
`/A`	Filtro: rimuovi solo file specifici attribuiti,
	usare il carattere `-` significa **non** attribuito come quel tipo.

Copia file senza xcopy

In questo esempio , l'utente BoeNoe ha mostrato come utilizzare il comando `xcopy` per copiare i file. C'è anche un comando aggiuntivo chiamato `copy` .

Qui è un semplice esempio:

```
copy foo.ext bar.ext
```

`foo.ext` su `bar.ext` e crea `bar.ext` quando non esiste. Possiamo anche specificare percorsi per il file, ma è sempre l'ideale per mettere virgolette (`"`) attorno ai percorsi, vedere qui per il motivo.

Ci sono anche molte bandiere disponibili per la `copy` , vedi `copy /?` o `help copy` su un prompt dei comandi per vedere di più.

Modifica di nth Line di un file

File batch non viene fornito con un metodo incorporato per la sostituzione di `n` esima riga di un file ad eccezione di `replace` e `append` (`>` e `>>`). Utilizzando `for` cicli, siamo in grado di emulare questo tipo di funzione.

```
@echo off
set file=new2.txt

call :replaceLine "%file%" 3 "stringResult"

type "%file%"
pause
exit /b

:replaceLine <fileName> <changeLine> <stringResult>
setlocal enableDelayedExpansion

set /a lineCount=%~2-1

for /f %%G in (%~1) do (
    if !lineCount! equ 0 pause & goto :changeLine
    echo %%G>>temp.txt
    set /a lineCount-=1
)

:changeLine
echo %~3>>temp.txt

for /f "skip=%~2" %%G in (%~1) do (
    echo %%G>>temp.txt
)

type temp.txt>%~1
del /f /q temp.txt

endlocal
exit /b
```

- Lo script principale chiama la funzione `replaceLine` , con il nomefile / quale riga modificare / e la stringa da sostituire.
- La funzione riceve l'input

    - Attraversa tutte le linee e le fa `echo` su un file temporaneo prima della linea di sostituzione
    - Fa `echo` alla linea di sostituzione del file
    - Continua a restituire il resto del file
    - Copia il file temporaneo nel file originale
    - E rimuove il file temporaneo.

- Lo script principale riprende il controllo e `type` il risultato.

Leggi Gestione file in file batch online: https://riptutorial.com/it/batch-file/topic/9253/gestione-file-in-file-batch

Capitolo 20: Ibridi Batch e JScript

introduzione

`JScript` è in realtà il superset di `Javascript` (è la versione 1.8.1 - quindi alcune funzionalità più recenti non sono disponibili) e possono essere incorporate in uno script `batch` per estendere le funzioni dello script `batch` . Di solito, le tecniche di incorporamento utilizzano le direttive JScript (non fanno parte dello standard Javascript ufficiale) al fine di separare il codice batch e JScript. JScript ti consente di lavorare con oggetti Com / ActiveX, oltre che con oggetti WMI oltre al Javascript standard.

Examples

JScript incorporato in un file batch

Questo esempio che segue è stato creato dall'utente Michael Dillon da questa risposta .

Considera il seguente script:

```
@set @junk=1 /*
@echo off
cscript //nologo //E:jscript %0 %*
goto :eof
*/

//JScript aka Javascript here
```

Questo frammento di script fa:

- Esegui il comando `cscript` che si chiama con tutti gli argomenti forniti.
- Come la parte dopo `@set @junk=1` è commentata (`/*` e `*/` Sono validi commenti JScript),
- JScript li ignorerà.
- **Nota: abbiamo bisogno della parte `@set @junk=1` perché il file batch non riconosce `/*` come un comando, ma un'istruzione `set` sarà una soluzione alternativa. JScript riconoscerà `/*` come commento in modo che l'altro file `batch` non venga eseguito dal motore JScript.**

Puoi aggiungere JScript dopo `*/` e iniziare ad estendere lo scripting dei file batch!

Esegui JScript con file temporanei

Come accennato qui , il metodo old-school per eseguire un altro script è utilizzando file

temporanei. È `echo` farlo `echo` in un file e quindi eseguirlo (e rimuoverlo facoltativamente).

Ecco il concetto di base:

```
@echo off
echo //A JS Comment > TempJS.js
echo //Add your code>>TempJS.js

cscript //nologo //e:cscript.exe TempJS.js

del /f /s /q TempJS.js
```

Ma questo richiederebbe molte istruzioni di `echo` per creare un JScript relativamente grande. Ecco un metodo migliore di Aacini.

```
@echo off
setlocal

rem Get the number of the "<resource>" line
for /F "delims=:" %%a in ('findstr /N "<resource>" "%~F0"') do set "start=%%a"

rem Skip such number of lines and show the rest of this file
(for /F "usebackq skip=%start% delims=" %%a in ("%~F0") do echo %%a) > TempJS.js

cscript //nologo //e:cscript.txt TempJS.js
del /f /s /q TempJS.js

goto :EOF

<resource>
JScript
JScript
JScript
```

Leggi Ibridi Batch e JSCript online: https://riptutorial.com/it/batch-file/topic/10578/ibridi-batch-e-jscript

Capitolo 21: Input e output reindirizzamento

Sintassi

- [comando] [[> | >> | <| 2> | 2 >>] file]
- [[> | >> | <| 2> | 2 >>] file] [comando]

Parametri

Parametro	Dettagli
comando	Qualsiasi comando valido.
`>`	Scrivi `STDOUT` su file.
`>>`	Aggiungi `STDOUT` al file.
`<`	Leggi il file su `STDIN` .
`2>`	Scrivi `STDERR` su file.
`2>>`	Aggiungi `STDERR` al file.
file	Il percorso di un file.

Osservazioni

- È possibile aggiungere tutti i reindirizzamenti desiderati, a condizione che il simbolo e il file di reindirizzamento rimangano insieme e nell'ordine corretto.

Examples

Un esempio...

```
@echo off
setlocal
set /p "_myvar=what is your name?"
echo HELLO!>file.txt
echo %_myvar%!>>file.txt
echo done!
pause
type file.txt
endlocal
exit
```

Ora file.txt ha il seguente aspetto:

```
HELLO!
John Smith!
```

(supponendo che tu abbia battuto `John Smith` come tuo nome).

Ora la console del file batch è simile a:

```
what is your name?John Smith
done!
Press any key to continue...
HELLO!
John Smith!
```

(e dovrebbe uscire così rapidamente che potresti non essere in grado di vedere nulla dopo il prompt `Press any key to coninue...`)

Reindirizza carattere speciale con espansione ritardata abilitata

Questo esempio fa eco al personaggio speciale `!` in un file. Funzionerebbe solo quando DelayedExpansion è disabilitato. Quando l'espansione ritardata è abilitata, dovrai utilizzare tre segni di omissione e un punto esclamativo come questo:

```
@echo off
setlocal enabledelayedexpansion


echo ^^^!>file
echo ^>>>file

goto :eof

    ^> is the text
    >> is the redirect operator

pause
endlocal
exit /b
```

Questo codice farà eco al testo seguente nel file

```
!
>
```

come

```
^^^ escapes the ! and echos it into the file
^> escapes the > and echos it into the file
```

Scrivi su un file

```
@echo off
```

```
echo Please input the file path, surrounded by "double quotation marks" if necessary.
REM If you don't want to redirect, escape the > by preceding it with ^
set /p filepath=^>

echo Writing a random number
echo %RANDOM% > %filepath%
echo Reading the random number
type %filepath%

REM Successive file writes will overwrite the previous file contents
echo Writing the current directory tree:
> %filepath% tree /A
echo Reading the file
type %filepath%

REM nul is a special file. It is always empty, no matter what you write to it.
echo Writing to nul
type %windir%\win.ini > nul
echo Reading from nul
type nul

echo Writing nul's contents to the file
type nul > %filepath%
echo Reading the file
type %filepath%
```

Leggi Input e output reindirizzamento online: https://riptutorial.com/it/batch-file/topic/7502/input-e-output-reindirizzamento

Capitolo 22: Macro di file batch

introduzione

In un prompt dei comandi, è possibile utilizzare DOSKEY per la creazione di macro. In un file batch è possibile definire una variabile che può essere chiamata come pezzo di codice e persino passargli argomenti.

Examples

Macro di base

Usando `DOSKEY` , possiamo creare macro per semplificare la digitazione di molti comandi nel prompt dei comandi. Dai un'occhiata al seguente esempio.

```
DOSKEY macro=echo Hello World
```

Ora se digiti `macro` nel prompt dei comandi, restituirà `Hello World` .

Commenti

Sfortunatamente, la macro `DOSKEY` non supporta i commenti, ma c'è una soluzione alternativa.

```
;= Comment
;= Comment
;= Remember to end your comment with ;=
;=
```

$ Usi di caratteri

Ci sono 3 usi del carattere `$` in una macro `DOSKEY` .

Comando separatore

`$T` è l'equivalente di `&` in uno script batch. Si possono unire i comandi insieme in questo modo.

```
DOSKEY test=echo hello $T echo world
```

Argomenti della riga di comando

Come `bash` (non `batch`), usiamo `$` per indicare l'argomento della riga di comando.

`$1` riferisce al primo argomento della riga di comando

`$2` riferisce al secondo argomento della riga di comando, ecc.

`$*` riferisce a tutti gli argomenti della riga di comando

Macro nello script batch

`DOSKEY` macro `DOSKEY` non funzionano in uno script batch. Tuttavia, possiamo usare un po 'di soluzione.

```
set DOSKEYMacro=echo Hello World
%DOSKEYMacro%
```

Questo script può simulare la funzione macro. Si può anche usare la e commerciale (`&`) per unire comandi, come `$T` in `DOSKEY` .

Se vuoi una "macro" relativamente grande, puoi provare una semplice funzione o dare un'occhiata ad altri argomenti di funzione qui .

Leggi Macro di file batch online: https://riptutorial.com/it/batch-file/topic/10791/macro-di-file-batch

Capitolo 23: Migliori pratiche

introduzione

Questo argomento si concentrerà sulle cose che si dovrebbero *(non obbligatorie)* fare in un file batch. L'utilizzo di queste "migliori pratiche" può migliorare l'effetto e la funzione di un file batch.

Examples

Citazioni

La maggior parte degli script batch online ha molti problemi di quotazione.

Esempi e soluzioni

Esempio A

```
if %var%==abc echo Test
```

Questo codice funziona - quando il contenuto di `%var%` non contiene spazio o altri caratteri speciali. Supponiamo che `%var%` contenga 1 spazio bianco. Ora `cmd.exe` vede:

```
if  ==abc echo Test
```

Ciò causerebbe un errore perché `cmd.exe` non capisce questa sintassi.

Soluzione A

```
if "%var%"=="abc" echo Test
```

Usando le virgolette, `cmd.exe` vede l'intera `%var%` (incluso spazio e caratteri speciali) come una sola stringa normale. Eppure questo non è il metodo di confronto più sicuro. Il più sicuro usa `echo` , `pipe` e `findstr` .

Esempio B

```
cd C:\User\Spaced Name\Spaced FileName.txt
```

`cd` cambierebbe solo la directory in `C:\User\Spaced` , poiché `cd` accetta solo un argomento di

per rso.

Soluzione B

Semplicemente aggiungendo virgolette sul percorso, il problema sarebbe risolto.

```
cd "C:\User\Spaced Name\Spaced FileName.txt"
```

Ci sono anche alcuni esempi che funzionano meglio usando le virgolette, come il `set /a` dichiarazione, ecc. Ma, quando si lavora su stringhe che contengono spazi o caratteri speciali, è di solito molto più sicuro usare le virgolette.

Codice degli spaghetti

Il codice Spaghetti indica un frammento di codice che utilizza molte e spesso confuse strutture. Come `GOTO` , eccezioni e codice incoerente.

Esempi e soluzioni

Esempio A

```
@echo off
set /a counter=0

:Loop
set /a counter=%counter% + 1
echo %counter%

if %counter% equ 10 goto :exit
goto :Loop

:exit
```

Questo programma viene fornito con un sacco di salti, ci rende più difficile sapere cosa sta facendo esattamente lo script.

Soluzione A

```
@echo off
for /l %%G in (0,1,10) echo %%G
```

Usando meno `GOTO` , abbiamo ridotto notevolmente la quantità di codice e possiamo concentrarci sul codice reale.

Esempio B

Considera le seguenti dichiarazioni.

```
:endGame
if %player1Score% gtr %player2Score% goto :player1wins
if %player1Score% lss %player2Score% goto :player2wins
goto :tie

:player1wins
echo player 1 wins
goto :eof

:player2wins
echo player 2 wins
goto :eof

:tie
echo tie
goto :eof
```

Questo snippet richiede molte istruzioni `goto` e può confondere il debug. Per semplificare queste affermazioni, possiamo usare il comando di `call` . Ecco lo script sopra in una condizione migliore.

```
:endGame
if %player1Score% gtr %player2Score% call :message player 1 wins
if %player1Score% lss %player2Score% call :message player 2 wins
if %player1Score% equ %player2Score% call :message tie

goto :eof

:message
echo %*
goto :eof
```

Entrambi gli script generano lo stesso risultato, ma il nuovo script è molto più breve e chiaro.

Leggi Migliori pratiche online: https://riptutorial.com/it/batch-file/topic/10746/migliori-pratiche

Capitolo 24: Per cicli in file batch

Sintassi

- per / l %% p in (startNumber, increment, endNumber) do comando
- per / f %% p in (nomefile) do comando
- per / f %% p in ("textStrings") comando do
- per / f %% p in ('comando') comando do
- per / r drive: \ path %% p nel comando (set) do
- per / d %% p nel comando (directory) do

Osservazioni

Il comando `for` accetta opzioni quando viene usato il flag `/f` . Ecco un elenco di opzioni che possono essere utilizzate:

- `delims=x` Delimitatore carattere (s) per separare token
- `skip=n` Numero di righe da saltare all'inizio di stringhe di file e di testo
- `eol=;` Carattere all'inizio di ogni riga per indicare un commento
- `tokens=n` Elementi numerati da leggere da ogni riga o stringa da elaborare
- `usebackq` Usa un altro stile di quotatura:

  Usa virgolette per nomi di file lunghi in "file"

  Utilizza le virgolette singole per "textStrings"

  Usa le virgolette posteriori per `comando`

Examples

Looping di ogni riga in un set di file

Quanto segue farà eco ad ogni riga nel file `C:\scripts\testFile.txt` . Le righe vuote non verranno elaborate.

```
for /F "tokens=*" %%A in (C:\scripts\testFile.txt) do (
  echo %%A
  rem do other stuff here
  )
```

L'esempio più avanzato mostra come il ciclo FOR derivato da un set di file limitato può essere utilizzato per reindirizzare l'esecuzione batch, salvando il contenuto cercato in un file:

```
@echo off
setlocal enabledelayedexpansion

for /f %%i in ('dir "%temp%\test*.log" /o:-d /t:w /b') do (
    set "last=%temp%\%%i"
    type !last! | find /n /i "Completed" >nul 2>&1 >> %temp%\Completed.log ^
     && (echo Found in log %%i & goto :end) || (echo Not found in log %%i & set "result=1"))

:: add user tasks code here
if defined result echo Performing user tasks...

:end
echo All tasks completed
exit /b
```

Nota, per quanto tempo le stringhe di comando sono divise in diverse code line e i gruppi di comandi sono separati da parentesi.

Visita in modo ricorsivo le directory in un albero delle directory

`for /r` comando `for /r` può essere utilizzato per visitare in modo ricorsivo tutte le directory in un albero di directory ed eseguire un comando.

```
@echo off
rem start at the top of the tree to visit and loop though each directory
for /r %%a in (.) do (
  rem enter the directory
  pushd %%a
  echo In directory:
  cd
  rem leave the directory
  popd
  )
```

Gli appunti:

- per / r - Loop through files (Recurse subfolders).
- pushd - Modifica la directory / cartella corrente e memorizza la cartella / percorso precedente per l'uso con il comando POPD.
- popd - Cambia la directory sul percorso / cartella più recente memorizzata dal comando PUSHD.

Rinominare tutti i file nella directory corrente

Quanto segue utilizza una variabile con un ciclo `for` per rinominare un gruppo di file.

```
SetLocal EnableDelayedExpansion

for %%j in (*.*) do (
  set filename=%%~nj
  set filename=!filename:old=new!
  set filename=Prefix !filename!
  set filename=!filename! Suffix
  ren "%%j" "!filename!%%~xj"
```

Definendo il nome della variabile `%%j` e associandolo a tutti i file correnti `(*.*)` , Possiamo usare la variabile in un ciclo `for` per rappresentare ogni file nella directory corrente.

Ogni iterazione (o passaggio) attraverso il ciclo elabora quindi un file diverso dal gruppo definito (che potrebbe essere stato ugualmente un qualsiasi gruppo, ad esempio `*.jpg` o `*.txt`).

Nel primo esempio, sostituiamo il testo: la stringa di testo "vecchio" è sostituita dalla stringa di testo "nuovo" (se, ma solo se, il testo "vecchio" è presente nel nome del file).

Nel secondo esempio, aggiungiamo del testo: il testo "Prefisso" viene aggiunto all'inizio del nome del file. Questa non è una sostituzione. Questa modifica verrà applicata a tutti i file nel gruppo.

Nel terzo esempio, di nuovo aggiungiamo del testo: il testo "Suffisso" viene aggiunto alla fine del nome del file. Ancora una volta, questa non è una sostituzione. Questa modifica verrà applicata a tutti i file nel gruppo.

La linea finale gestisce effettivamente la ridenominazione.

Iterazione

```
for /L %%A in (1,2,40) do echo %%A
```

Questa riga verrà iterata da 1 a 39, aumentando di 2 ogni volta.

Il primo parametro, `1` , è il numero iniziale.

Il secondo parametro, `2` , è l'incremento.

Il terzo parametro, `40` , è il massimo.

Leggi Per cicli in file batch online: https://riptutorial.com/it/batch-file/topic/3695/per-cicli-in-file-batch

Capitolo 25: Se affermazioni

Sintassi

- if [/ i] StringToCompare1 == StringToCompare2 (commandA) else (commandB)
- if errorlevel 1 (commandA) else (commandB)
- if% errorlevel% == 1 (commandA) else (commandB)
- se esiste Nome file (comandoA) altro (comandoB)
- se definito VariableName (commandA) else (commandB)

Osservazioni

Ci sono alcune sintassi tra cui scegliere in un'istruzione `if` . Useremo `if string1==string2` come esempio.

Sintassi a 1 linea

- `if string1==string2 commandA`
- `if string1==string2 (commandA)`

- `if string1==string2 (commandA) else (commandB)`
- `if string1==string2 (commandA) else commandB`

- `if string1==string2 (commandA)else (commandB)`
- `if string1==string2 (commandA)else commandB`

Sintassi multilinea

```
if string1==string2 (
    commandA
)
```

O

```
if string1==string2 (
    commandA
) else (
    commandB
)
```

Sono ancora disponibili alcune sintassi aggiuntive.

Examples

Confronto dei numeri con la dichiarazione IF

```
SET TEST=0

IF %TEST% == 0 (
    echo TEST FAILED
) ELSE IF %TEST% == 1 (
    echo TEST PASSED
) ELSE (
    echo TEST INVALID
)
```

Confronto tra stringhe

```
IF "%~1" == "-help" (
    ECHO "Hello"
)
```

dove `%1` riferisce al primo argomento della riga di comando e `~` rimuove tutte le virgolette che sono state incluse quando è stato chiamato lo script.

Confronto di Errorlevel

```
If Errorlevel 1 (
    Echo Errorlevel is 1 or higher

    REM The phrase "1 or higher" is used because If Errorlevel 1 statement means:
    REM                                          If %Errorlevel% GEQ 1
    REM                                    Not If %Errorlevel% EQU 1
)
```

o

```
If "%Errorlevel%"=="1" (
    Echo Errorlevel is 1
)
```

Lo script sopra controllerebbe la variabile Errorlevel (built-in). L'operatore `not` può essere usato.

```
Set "Test=%Errorlevel%"

If "%Test%" == "1" (
    Echo Errorlevel is 1
)
```

Anche questo funziona.

Si noti che alcuni comandi **non influiscono sul livello di errore** :

Rompere
- Eco
- endlocal
- Per
- Se
- Pausa
- Rem
- Rd / Rmdir
- Impostato
- Titolo

I seguenti comandi **impostano ma non cancellano errorlevel** :

- Cls
- Vai a
- chiavi
- popd
- Cambio

I seguenti comandi **impostano i codici di uscita ma non il livello di errore** :

- Rd / Rmdir

I seguenti comandi **impostano errorlevel ma non i codici di uscita** :

- MD / Mkdir

Controlla se il file esiste

```
If exist "C:\Foo\Bar.baz" (
    Echo File exist
)
```

Questo controlla se l'esistenza del file C: \ Foo \ Bar.baz. Se questo esiste, echos File esiste Anche l'operatore `Not` può essere aggiunto.

Se la variabile esiste / imposta

```
If Defined Foo (
    Echo Foo is defined
)
```

Questo controllerebbe se una variabile è definita o meno. Di nuovo, può essere usato l'operatore `Not` .

Leggi Se affermazioni online: https://riptutorial.com/it/batch-file/topic/5475/se-affermazioni

Capitolo 26: Stack di directory

Sintassi

- PUSHD [percorso]
- POPD

Parametri

Parametro	Dettagli
sentiero	La directory per navigare

Osservazioni

- Usando `pushd` senza parametri verrà stampato lo stack.
- Il comando `popd` sovrascriverà il valore corrente della directory corrente.

Examples

Elimina file di testo

L'esempio seguente mostra come utilizzare il comando `pushd` e il comando `popd` in un programma batch per modificare la directory corrente da quella in cui è stato eseguito il programma batch e quindi modificarla di nuovo:

```
@echo off
rem This batch file deletes all .txt files in a specified directory
pushd %1
del *.txt
popd
cls
echo All text files deleted in the %1 directory
```

Sourced from https://technet.microsoft.com/en-us/library/cc771180%28v=ws.11%29.aspx

Stampa stack di directory

Per stampare lo stack di directory, usa il comando `pushd` senza parametri:

```
@echo off

cd C:\example\
pushd one
pushd ..\two
```

```
pushd ..\..


pushd
echo Current Directory: %cd%

echo:
popd
pushd three

pushd
echo Current Directory: %cd%
```

Produzione:

```
C:\example\two
C:\example\one
C:\example
Current Directory: C:\

C:\example\two
C:\example\one
C:\example
Current Directory: C:\example\two\three
```

Leggi Stack di directory online: https://riptutorial.com/it/batch-file/topic/4288/stack-di-directory

Capitolo 27: Usando Goto

introduzione

Goto è semplice. Utilizzando semplici istruzioni goto, puoi spostarti ovunque nel codice. Può anche essere usato per creare funzioni (mostrato in come creare funzioni).

Sintassi

- goto: etichetta
- vai all'etichetta
- goto: EOF

Parametri

Parametro	Dettagli
`:Label`	Qualsiasi etichetta valida (definita da `:<LabelName>`)
`:EOF`	Un'etichetta predefinita che esce dallo script corrente della funzione (come `exit /b`)

Osservazioni

Quindi, in altre parole, se il numero inserito dal giocatore è 1, tornerà alla parte Nome del codice.

quindi se l'input è uguale a 1, torna alla riga con: Nome

Assicurarsi che se si utilizza questo, la parola inizia con il Colen (:).

Examples

Programmi di esempio

Per esempio:

```
echo Hello!
pause >nul
:Name
echo What Is Your Name
set /p Input=Name:
echo so %Input% Is Your Name, right?
echo Rename?
echo 1 For Yes
echo 2 For No
```

```
set /p Input=Rename:
if %Input%=1 goto Name
```

Un altro esempio:

```
@echo off
echo 1 or 2?
set /p input=Choice:
if %input%=1 goto Skip
echo You Chose 1
pause >nul
echo So time for stuff
pause >nul
echo Random Stuf
pause >nul
:Skip
echo So that's it.
pause >nul
```

Vai con variabile

`Goto` accetta l'uso del valore variabile per fungere da etichetta per goto.

Esempio:

```
@echo off

echo a = 1
echo b = 2

set /p "foo=Enter option:"
goto %foo%
```

Tuttavia, dovresti controllare l'input in modo che non vada da qualche parte che non esiste. Passare a un'etichetta indefinita interromperà immediatamente lo script batch.

Leggi Usando Goto online: https://riptutorial.com/it/batch-file/topic/9164/usando-goto

Capitolo 28: Variabili nei file batch

Examples

Dichiarazione

Per creare una variabile semplice e assegnarla a un valore o una stringa, utilizzare il comando `SET` :

```
SET var=10
```

Qui, il codice dichiara una nuova variabile `var` con un valore di `10` . Di default tutte le variabili sono memorizzate internamente come stringhe; questo significa che il valore `10` non è diverso da `foo1234` o `Hello, World!`

Note sulle virgolette

Le virgolette utilizzate saranno incluse nel valore della variabile:

```
SET var="new value"             <-- %var% == '"new value"'
```

Spazi nelle variabili

Il linguaggio batch considera gli spazi come parti accettabili dei nomi di variabili. Ad esempio, `set var = 10` darà come risultato una variabile chiamata `var` che contiene il valore `10` (notare lo spazio extra a destra di var e il lato sinistro del 10).

Usare le virgolette per eliminare gli spazi

Per evitare spazi, utilizzare le virgolette per l'intero compito; il nome e il valore della variabile. Questo impedisce anche spazi accidentali rovesciata alla fine della linea (la ␣ carattere denota uno spazio):

```
SET␣var=my␣new␣value␣        <-- '%var%' == 'my new value '
SET␣"var=my␣new␣value"␣       <-- '%var%' == 'my new value'
```

Inoltre, utilizzare le virgolette quando si uniscono più istruzioni con `&` o `|` - In alternativa, metti il simbolo direttamente dopo la fine del valore della variabile:

```
SET var=val & goto :next        <-- '%var%' == 'val '
SET "var=val" & goto :next      <-- '%var%' == 'val'
SET var=val& goto :next         <-- '%var%' == 'val'
```

```
echo %var%
```

Questo codice farà eco al valore di `var`

Se viene utilizzato `setLocal EnableDelayedExpansion` , quanto segue `setLocal EnableDelayedExpansion` il valore di `var` (l'espressione standard% var% non funzionerà in quel contesto).

```
echo !var!
```

Nei file batch, le variabili possono essere utilizzate in qualsiasi contesto, incluse parti di comandi o parti di altre variabili. Non puoi chiamare una variabile prima di definirla.

Uso delle variabili come comandi:

```
set var=echo
%var% This will be echoed
```

Uso delle variabili in altre variabili:

```
set var=part1
set %var%part2=Hello
echo %part1part2%
```

Sostituzione variabile

A differenza di altri linguaggi di programmazione, in un file batch una variabile viene sostituita dal suo valore reale **prima** dell'esecuzione dello script batch. In altre parole, la sostituzione viene eseguita quando lo script viene *letto* in memoria dal processore dei comandi, non quando lo script viene *eseguito* successivamente.

Ciò consente l'uso di variabili come comandi all'interno dello script e come parte di altri nomi di variabili nello script, ecc. Lo "script" in questo contesto è una linea - o un blocco - di codice, circondato da parentesi tonde: `()` .

Ma questo comportamento significa che non è possibile modificare il valore di una variabile all'interno di un blocco!

```
SET VAR=Hello
FOR /L %%a in (1,1,2) do (
    ECHO %VAR%
    SET VAR=Goodbye
)
```

stamperà

```
Hello
Hello
```

poi è (come vedi, quando si guarda lo script eseguito nella finestra di comando) viene valutato per:

```
SET VAR=Hello
FOR /L %%a in (1,1,2) do (
    echo Hello
    SET VAR=Goodbye
)
```

Nell'esempio precedente, il comando `ECHO` viene valutato come `Hello` quando lo script viene letto in memoria, quindi lo script farà eco a `Hello` per sempre, tuttavia molti passaggi vengono fatti attraverso lo script.

Il modo per ottenere il comportamento più "tradizionale" delle variabili (della variabile che viene espansa mentre lo script è in esecuzione) è di abilitare "l'espansione ritardata". Ciò comporta l'aggiunta di quel comando nello script prima dell'istruzione loop (di solito un ciclo FOR, in uno script batch) e l'utilizzo di un punto esclamativo (!) Anziché di un segno di percentuale (%) nel nome della variabile:

```
setlocal enabledelayedexpansion
SET VAR=Hello
FOR /L %%a in (1,1,2) do (
    echo !VAR!
    SET VAR=Goodbye
)
endlocal
```

stamperà

```
Hello
Goodbye
```

La sintassi `%%a in (1,1,2)` fa sì che il ciclo venga eseguito 2 volte: nella prima occasione, la variabile porta il suo valore iniziale di "Ciao", ma al secondo passaggio attraverso il ciclo - dopo aver eseguito il secondo SET dell'istruzione come ultima azione al primo passaggio - questo è cambiato nel valore modificato 'Arrivederci'.

Sostituzione variabile avanzata

Ora, una tecnica avanzata. L'utilizzo del comando `CALL` consente al processore di comandi batch di espandere una variabile situata sulla stessa riga dello script. Questo può fornire un'espansione multilivello, ripetendo l'uso di CALL e modificatori.

Questo è utile, ad esempio, in un ciclo FOR. Come nel seguente esempio, dove abbiamo un elenco numerato di variabili:

```
"c:\MyFiles\test1.txt" "c:\MyFiles\test2.txt" "c:\MyFiles\test3.txt"
```

Possiamo ottenere questo utilizzando il seguente ciclo FOR:

```
setlocal enabledelayedexpansion
```

```
f %%x in (%*) do (
    set /a "i+=1"
    call set path!i!=%%~!i!
    call echo %%path!i!%%
)
endlocal
```

Produzione:

```
c:\MyFiles\test1.txt
c:\MyFiles\test2.txt
c:\MyFiles\test3.txt
```

Si noti che la variabile `!i!` viene prima esteso al suo valore iniziale, 1, quindi la variabile risultante,% 1, viene espansa al suo valore effettivo di `c:\MyFiles\test1.txt` . Questa è la *doppia espansione* della variabile `i` . Nella riga successiva, `i` viene nuovamente espanso in due, utilizzando il comando `CALL ECHO` insieme al prefisso variabile `%%` , quindi stampato sullo schermo (ovvero visualizzato sullo schermo).

Su ogni passaggio successivo attraverso il ciclo, il numero iniziale viene aumentato di 1 (a causa del codice `i+=1`). Così aumenta a `2` al `2` ° passaggio attraverso il ciclo ea `3` al `3` ° passaggio. Così la stringa echeggiata sullo schermo cambia con ogni passaggio.

Dichiarare più variabili

Quando vengono definite più variabili all'inizio del batch, è possibile utilizzare un modulo di definizione breve utilizzando una stringa sostitutiva .

```
@echo off
set "vars=_A=good,_B=,_E=bad,_F=,_G=ugly,_C=,_H=,_I=,_J=,_K=,_L=,_D=6
set "%vars:,=" & set "%"

for /f %%l in ('set _') do echo %%l
exit /b

_A=good
_D=6
_E=bad
_G=ugly
```

Nota nell'esempio precedente, le variabili sono ordinate alfabeticamente in ordine alfabetico, quando stampate sullo schermo.

Utilizzo di una variabile come matrice

È possibile creare un insieme di variabili che possono agire in modo simile a un array (sebbene non siano un vero oggetto array) utilizzando gli spazi `SET` :

```
@echo off
SET var=A "foo bar" 123
for %%a in (%var%) do (
    echo %%a
```

```
)
echo Get the variable directly: %var%
```

Risultato:

```
A
"foo bar"
123
Get the variable directly: A "foo bar" 123
```

È anche possibile dichiarare la variabile utilizzando gli indici in modo da poter recuperare informazioni specifiche. Questo creerà più variabili, con l'illusione di un array:

```
@echo off
setlocal enabledelayedexpansion
SET var[0]=A
SET var[1]=foo bar
SET var[2]=123
for %%a in (0,1,2) do (
    echo !var[%%a]!
)
echo Get one of the variables directly: %var[1]%
```

Risultato:

```
A
foo bar
123
Get one of the variables directly: foo bar
```

Si noti che nell'esempio sopra, non è possibile fare riferimento a `var` senza dichiarare quale sia l'indice desiderato, perché `var` non esiste per conto proprio. In questo esempio viene utilizzata anche l' `setlocal enabledelayedexpansion` congiuntamente ai punti esclamativi di `!var[%%a]!` . È possibile visualizzare ulteriori informazioni a riguardo nella Documentazione Ambito di sostituzione variabile .

Operazioni su variabili

```
set var=10
set /a var=%var%+10
echo %var%
```

Il valore finale di `var` è 20.

La seconda riga non funziona all'interno di un blocco di comando utilizzato ad esempio su una condizione **IF** o su un ciclo **FOR in** quanto sarebbe necessaria un'espansione ritardata invece dell'espansione della variabile di ambiente standard.

Ecco un altro modo migliore di lavorare anche in un blocco di comandi:

```
set var=10
```

```
set /A var+=10
echo %var%
```

L'ambiente del prompt dei comandi supporta i valori interi a 32 bit con segno:

- aggiunta `+` e `+=`
- sottrazione `-` e `-=`
- moltiplicazione `*` e `*=`
- divisione `/` e `/=`
- modulo divisione `%` e `%=`
- bit a bit AND `&`
- bit a bit OR `|`
- bit a bit NOT `~`
- XOR bit a bit `^`
- spostamento a sinistra bit a bit `<<`
- spostamento a destra bit a bit `>>`
- NOT logico `!`
- unario meno `-`
- raggruppamento con `(` e `)`

L'interprete dei comandi di Windows non supporta i valori integer a 64 bit o i valori in virgola mobile nelle espressioni aritmetiche.

Nota: l'operatore `%` deve essere scritto in un file batch come `%%` per essere interpretato come operatore.

In una finestra del prompt dei comandi che esegue la riga di comando `set /A Value=8 % 3` assegna il valore `2` alla variabile di ambiente `Value` e inoltre le uscite `2` .

In un file batch deve essere scritto `set /A Value=8 %% 3` per assegnare il valore `2` alla variabile di ambiente `Value` e nulla viene emesso rispettivamente scritto per gestire **STDOUT** (output standard). Un `set /A Value=8 % 3` righe `set /A Value=8 % 3` in un file batch comporterebbe un messaggio di errore *Operatore mancante* all'esecuzione del file batch.

L'ambiente richiede lo switch `/A` per le operazioni aritmetiche, non per le variabili stringa ordinarie.

Ogni stringa nell'espressione aritmetica dopo che `set /A` indica se un numero o un operatore viene interpretato automaticamente come nome di una variabile di ambiente.

Per questo motivo, fare riferimento al valore di una variabile con `%variable%` o con `!variable!` non è necessario quando il nome della variabile consiste solo di caratteri di parole (0-9A-Za-z_) con il primo carattere che non è una cifra che è particolarmente utile all'interno di un blocco di comandi che inizia con `(` e termina con una corrispondenza `)` .

I numeri vengono convertiti da stringa a intero con la funzione Ct C ++ strtol con `base` zero, il che significa determinazione della base automatica che può facilmente portare a risultati imprevisti.

Esempio:

```
set Divided=11
set Divisor=3

set /A Quotient=Divided / Divisor
set /A Remainder=Divided %% Divisor

echo %Divided% / %Divisor% = %Quotient%
echo %Divided% %% %Divisor% = %Remainder%

set HexValue1=0x14
set HexValue2=0x0A
set /A Result=(HexValue1 + HexValue2) * -3

echo (%HexValue1% + %HexValue2%) * -3 = (20 + 10) * -3 = %Result%

set /A Result%%=7
echo -90 %%= 7 = %Result%

set OctalValue=020
set DecimalValue=12
set /A Result=OctalValue - DecimalValue

echo %OctalValue% - %DecimalValue% = 16 - 12 = %Result%
```

L'output di questo esempio è:

```
11 / 3 = 3
11 % 3 = 2
(0x14 + 0x0A) * -3 = (20 + 10) * -3 = -90
-90 %= 7 = -6
020 - 12 = 16 - 12 = 4
```

Le variabili non definite sulla valutazione dell'espressione aritmetica sono sostituite con il valore 0.

Impostazione delle variabili da un input

Utilizzando l'opzione `/p` con il comando `SET` è possibile definire le variabili da un input.

Questo input può essere un input dell'utente (tastiera):

```
echo Enter your name :
set /p name=
echo Your name is %name%
```

Quale può essere semplificato in questo modo:

```
set /p name=Enter your name :
echo Your name is %name%
```

Oppure puoi ottenere l'input da un file:

```
set /p name=< file.txt
```

in questo caso otterrai il valore della prima riga da `file.txt`

Ottenere il valore di varie linee in un file:

```
(
   set /p line1=
   set /p line2=
   set /p line3=

) < file.txt
```

Leggi Variabili nei file batch online: https://riptutorial.com/it/batch-file/topic/3528/variabili-nei-file-batch

ritoli di coda

S. No	Capitoli	Contributors
1	Iniziare con il file batch	BoeNoe, Clijsters, Community, DavidPostill, Derpcode, geisterfurz007, Jeffrey Lin, loadingnow, Mee, rudicangiotti, sambul35, Scott Beeson, Sourav Ghosh, stark, SteveFest, ths
2	Aggiungi ritardo al file batch	SteveFest, Tearzz, wizzwizz4
3	Argomenti della riga di comando del file batch	DavidPostill, LotPings, npocmaka, sambul35
4	Batch e ibridi VBS	npocmaka, SteveFest
5	Bug nel processore cmd.exe	SteveFest, X. Liu
6	Bypassare le limitazioni aritmetiche nei file batch	npocmaka
7	Cambiare le directory e elencarne i contenuti	Aravind .KEN, SomethingDark, SteveFest, wizzwizz4
8	Casuale nei file batch	SteveFest
9	Cerca stringhe in batch	sambul35
10	Comandi batch deprecati e loro sostituzioni	npocmaka, SteveFest
11	Commenti nei file batch	0x90h, BoeNoe, DavidPostill, Gábor Bakos, JosefZ, LotPings, SachaDee, SomethingDark, Sourav Ghosh, Stephan, SteveFest, wasatchwizard
12	Creazione di file tramite batch	Aravind .KEN, David Starkey, fruitSalad266, J03L, LeoDog896, loadingnow, Tearzz

13	Differenze tra Batch (Windows) e Terminal (Linux)	SomethingDark, SteveFest
14	Eco	DavidPostill, fruitSalad266, Keith Hall, npocmaka, SomethingDark, Tearzz, wizzwizz4
15	Elevati privilegi nei file batch	DavidPostill, elzooilogico, sambul35
16	Escaping caratteri speciali	npocmaka, SteveFest
17	File batch e ibridi PowerShell	npocmaka, SteveFest
18	funzioni	ender_scythe, npocmaka, SteveFest
19	Gestione file in file batch	BoeNoe, LeoDog896, SteveFest
20	Ibridi Batch e JSCript	npocmaka, SteveFest
21	Input e output reindirizzamento	cascading-style, SomethingDark, SteveFest, wizzwizz4
22	Macro di file batch	SteveFest
23	Migliori pratiche	SteveFest
24	Per cicli in file batch	David Starkey, DavidPostill, Ed999, Mee, sambul35, SomethingDark, SteveFest
25	Se affermazioni	npocmaka, Ozair Kafray, SomethingDark, SteveFest
26	Stack di directory	DavidPostill, wizzwizz4
27	Usando Goto	LeoDog896, SteveFest
28	Variabili nei file batch	DavidPostill, Ed999, Jay, Jeffrey Lin, Mee, Mofi, SachaDee, sambul35, SomethingDark, SteveFest, Tearzz, ths, Toomaja, wasatchwizard, wizzwizz4