批次指令

請選取指令以取得相關資訊。

call	goto	setlocal
echo	if	shift
endlocal	pause	%
for	rem	

批次程式 (也稱為批次檔) 可簡化例行或重複性高的工作。「批次程式」是未經格式化的文字檔，其內含一至多項的指令，並會以 .bat 或 .cmd 作為副檔名。當在命令提示字元上鍵入該檔名時，檔案中的指令便會依序執行。

批次檔中可以包含任何指令。而 for、goto 及 if 指令更可讓您在批次檔中，有條件地處理指令。例如，if 指令可根據條件的結果執行指令。其他的指令則可讓您控制輸入、輸出及呼叫其他的批次程式。至於可替換的參數 %，則可擴充批次指令檔的引數變數。

% (可取代的參數)

可取代的參數 %0 及 %1 到 %9 可放在批次檔中的任何位置。執行批次檔時，會用批次檔名稱取代 %0，而用指令行中所輸入的對應參數來取代 %1 到 %9 的引數變數。

例如，若要將某個資料夾中的內容複製到另一個資料夾中，則需在批次檔中新增下列敘述式：

xcopy %1*.* %2

執行該檔案時，需鍵入下列指令：

mybatch.bat C:\afolder D:\bfolder。

其作用與您將 xcopy C:\afolder *.* D:\bfolder 寫入批次檔中的作用相同。

% 參數會擴充批次指令檔引數變數 (%0、%1、 ...、%9) (如下所示)：

批次指令檔中的 %* 是參照所有引數的萬用字元。獨立引數變數的擴充選項會在下列表格中予以說明。

變數	描述
%~1	擴充 %1 並移除用來括住它的引號 (")
%~f1	將 %1 擴充為完整的路徑名稱
%~d1	將 %1 擴充為磁碟機代號
%~p1	將 %1 擴充為路徑
%~n1	將 %1 擴充為檔名
%~x1	將 %1 擴充為副檔名
%~s1	展開的路徑只包含短檔名
%~a1	將 %1 擴充為檔案屬性
%~t1	將 %1 擴充為檔案的日期/時間
%~z1	將 %1 擴充為檔案大小
%~$PATH:1	搜尋 PATH 環境變數中所列的目錄，並將 %1 擴充為第一個找到之目錄的完整名稱。如果未定義環境變數名稱，或是找不到檔案，則會將這個輔助字元擴充成空字串。

這些輔助字元可合併使用，以取得複合結果：

變數	描述
%~dp1	將 %1 擴充為磁碟機代號及路徑
%~nx1	將 %1 擴充為檔名及副檔名
%~dp$PATH:1	在 PATH 環境變數所列的目錄中搜尋 %1，並將其擴充為第一個找到之目錄的磁碟機代號和路徑
%~ftza1	將 %1 擴充與 dir 類似的輸出行

在上述範例中，可用其他有效值取代 %1 及 PATH。有效的引數編號必須以 %~ 語法結尾。%~ 輔助字元可能無法與 %* 搭配使用。

Rem

可讓您在批次檔或設定檔案中包含註解或說明。

rem [*comment*]

參數

comment

指定要當成註解的字串

附註

使用 echo 指令來顯示註解

rem 指令不會在螢幕上顯示指令。您必須在批次檔或 Config.nt 檔案中使用 echo on 指令，才能在螢幕上顯示註解。

批次檔註解的限制

您無法在批次檔註解中使用重新導向字元 (" 或 ") 或管線 (|)。

使用 rem 以新增垂直間距

可使用不含註解的 rem 將垂直間距新增到批次檔中，但您也可使用空白行。Windows 2000 在處理批次程式時會略過空白行。

範例

下例顯示使用說明及垂直間距來加上註解的批次檔：

```
@echo off
rem This batch program formats and checks new disks.
rem It is named Checknew.bat.
rem
```

```
echo Insert new disk in drive B.
pause
format b:/v
chkdsk b:
```

假設您要在 **prompt** 指令之前的 Config.nt 檔案中包含說明註解。若要做這樣的處理，請將下列數行新增到 Config.nt 中：

```
rem Set prompt to indicate current directory
prompt $p$g
```

If

在批次程式中執行條件處理。如果 **if** 指令中所指定的條件為真，Windows 2000 會完成於條件後的指令。如果條件為偽，Windows 2000 會略過 **if** 中的指令，而執行 **else** 中所指定的指令。

if [**not**] **errorlevel** *number command* [**else** *expression*]

if [**not**] *string1*==*string2 command* [**else** *expression*]

if [**not**] **exist** *filename command* [**else** *expression*]

啟用指令擴充：

if [**/i**] *string1 compare-op string2 command* [**else** *expression*]

if cmdextversion *number command* [**else** *expression*]

if defined *variable command* [**else** *expression*]

參數

not

指定 Windows 2000 僅能在此條件為偽時才可完成該指令。

errorlevel *number*

僅在透過 Cmd.exe 執行的程式傳回之結束碼大於或等於 *number* 時，才將條件指定為真。

command

指定需符合先前的條件，Windows 2000 才能完成指令。

string1==*string2*

僅當 *string1* 及 *string2* 相同時，條件才為真。這些值可以是文字字串或批次變數 (如 *%1*)。文字字串不需要引號。

exist *filename*

如果 *filename* 存在，則條件為真。

compare-op

下列由三個字母表示的其中一種比較運運算元：

運運算元　　描述

EQU　　等於

NEQ　　不等於

LSS　　小於

LEQ　　小於或等於

GTR　　大於

GEQ　　大於或等於

/i

指定 **/i** 參數時，會強制字串忽略大小寫的比較。**/i** 參數也適用於 **if** 中的 *string1==string2* 格式。這是普通的比較，其中如果 *string1* 及 *string2* 都是全由數字組成，則會將字串轉換成數字並進行數值比較。

cmdextversion *number*

除了與 Cmd.exe 的「指令擴充」功能版本編號相比較以外，**cmdextversion** 條件的運作方式與 *errorlevel* 類似。第一個版本為 1。對此指令擴充的重大增強會將此編號加一。停用指令擴充時，**cmdextversion** 條件絕不會為真。

defined *variable*

如果定義了環境變數，則 **defined** (除了使用環境變數名稱並傳回真值外) 條件的運作方式與 **exist** 指令類似。此條件新增了三個變數：**%errorlevel%**、**%cmdcmdline%** 及 **%cmdextversion%**。

%errorlevel% 會擴充到目前 *errorlevel* 的字串呈現中；在此字串呈現中有一個名為 ERRORLEVEL 的環境變數，可讓您得到其值。執行程式後，下列說明瞭 **errorlevel** 的使用：

```
    goto answer%erorlevel%
:answer0
echo Program had return code 0
```

```
:answer1
echo Program had return code 1
```

您也可以使用列在 *compare-op* 上的比較運算元：

```
    if %errorlevel% LEQ 1 goto okay
```

%cmdcmdline% 會擴充到優先於任何透過 Cmd.exe 處理的原始指令行；這些指令行中沒有名為 **cmdcmdline** 的環境變數，可讓您取得其值。

%cmdextversion% 會擴充到對目前 **cmdextversion** 值的字串呈現中；而在這些字串中並沒有名為 CMDEXTVERSION 的環境變數，可讓您取得其值。

expression

在 **else** 子句中，*expression* 是由 Windows 2000 指令和任何傳給該指令的參數所組成。

附註

程式停止時，會將結束碼傳回給 Windows 2000。*errorlevel* 參數可讓您將此結束碼當成條件使用。

範例

使用 if 以驗證檔案的存在。

如果 Windows 2000 找不到檔案 Product.dat，會出現下列訊息：

```
  if not exist product.dat echo Can't find data file
```

發生錯誤時，會使用 if 公佈訊息。

下例顯示如果在格式化磁碟機 A: 中的磁碟期間發生錯誤的錯誤訊息：

```
  :begin
@echo off
```

```
format a:/s
if not errorlevel 1 goto end
echo An error occurred during formatting.
:end
echo End of batch program.
```

如果未發生錯誤，則會跳過錯誤訊息。

使用 if 可驗證目錄是否存在。

下例會測試目錄是否存在。**if** 指令無法直接測試目錄，而空的 (NUL) 裝置是存在於每個目錄中。因此，您可以測試空的裝置，以決定目錄是否存在。

```
  if exist c:mydir\nul goto process
```

使用 else 子句

else 子句必須發生在 **if** 指令後的同一行上。例如：

```
    IF EXIST filename. (
del filename.
) ELSE (
echo filename. missing.
    )
```

下列並不會運作，因為 **del** 指令必須以新行終止：

```
    IF EXIST filename. del filename.ELSE echo filename. missing
```

下列並不會運作，因為 **else** 指令必須與 **if** 指令結尾位元於同一行。

```
    IF EXIST filename. del filename.
ELSE echo filename. missing
```

如果想要格式化所有在單一行的下列原始敘述式格式，則其運作：

```
    IF EXIST filename.(del filename.)ELSE echo filename. missing
```

For

對一組檔案中的每一份檔案執行指定的指令。

您可以在批次程式中或直接在命令提示字元中使用 for 指令。

若要在批次程式中使用 for，請使用下列語法：

for %%*variable* in (*set*) do *command* [*command-parameters*]

若要在命令提示字元中使用 for，請使用下列語法：

for %*variable* in (*set*) do *command* [*command-parameters*]

參數

%%*variable* 或 %*variable*

表示可替換的參數。for 指令會以指定之 *set* 中的各個文字串，取代 %%*variable* (或 %*variable*)，直到指令 (*command-parameters* 中所指定) 處理完所有的檔案為止。使用 %%*variable* 完成批次程式中的 for 指令。使用 %*variable* 從命令提示字元完成 for 指令。變數名稱需區分大小寫。

(*set*)

指定要以特定之指令處理的檔案或文字串。必須使用括號。

指令

指定要在特定之 *set* 中所含的各個檔案中完成的指令。

command-parameters

指定要與特定之指令 (如果指定的指令使用了參數) 搭配使用的任何參數。

啟用指令延伸 (Windows 2000 的預設設定) 即可支援其他格式的 for 指令。

其他格式的 For 指令

啟用指令延伸可支援下列格式的 for 指令：

只針對目錄

for /D [%% | %]*variable* in (*set*) do *command* [*command-parameters*]

如果 *set* 含有萬用字元 (* 及 ?)，則比對作業會針對目錄名稱，而不是檔案名稱。

遞迴

for /R [[*drive* :]*path*] [%% | %]*variable* in (*set*) do *command* [*command-parameters*]

從 [*drive*:]*path* 所指定的樹狀目錄開始，在其下的每一個目錄中執行 for 敘述式。如果 /R 之後未指定任何目錄，則便會以目前的目錄為其設定。如果 *set* 中只有一個句點 (.) 字元，其便會列舉樹狀目錄。

反覆

for /L [%% | %]*variable* in (*start*,*step*,*end*) do *command* [*command-parameters*]

set 是指從開始到結束期間，依級數遞增或遞減的序號。因此 (1,1,5) 的順序即是 1 2 3 4 5，(5,-1,1) 的順序則為 (5 4 3 2 1)。

檔案剖析

for /F ["*options*"] [%% | %]*variable* in (*filenameset*) do *command* [*command-parameters*]

for /F ["*options*"] [%% | %]*variable* in ("*literal string*") do *command* [*command-parameters*]

for /F ["*options*"] [%% | %]*variable* in ('*command*') do *command* [*command-parameters*]

倘若您使用了 usebackq 選項：

for /F ["*options*"] [%% | %]*variable* in (*filenameset*) do *command* [*command-parameters*]

for /F ["*options*"] [%% | %]*variable* in ('*literal string*') do *command* [*command-parameters*]

for /F ["*options*"] [%% | %]*variable* in (`*command*`) do *command* [*command-parameters*]

filenameset 參數可指定檔案名稱。在開始執行下一份 *filenameset* 所指定的檔案之前，皆會開啟、讀取並處理各份檔案。

處理程式包括讀取檔案、將檔案分成獨立的文字行，以及將每一行剖析成零或多個 Token Ring。之後則會利用設定給找到之 Token Ring 字串的變數值，呼叫 for 迴圈的主體。預設 /F 會從每一份檔案傳遞各行的第一個獨立空白 Token Ring。

跳過空白行。藉由指定可省略的 options 參數，可以覆寫預設的剖析行為。此為引號字串，其內含有一或多個用以指定不同剖析選項的關鍵字。這些關鍵字包括：

關鍵字	描述
eol=c	指定行尾的註釋字元 (僅一個字元)
skip=n	指定要從檔案開頭開始跳過的行數。
delims=xxx	指定分隔字元集合。此會取代預設的空格及 tab 鍵分隔字元集合。
tokens=x,y,m-n	指定各行中要反覆傳遞給 for 主體的 Token Ring。此項指定會導致額外的變數名稱配置。*m-n* 代表範圍，可指定第 *m* 個到第 *n* 個 Token Ring。如果 tokens= string 的最後一個字元是星號，便會再額外配置一個變數，並會在剖析了最後一個 Token Ring 之後，接收行內的剩餘文字。
usebackq	指定將反括號字串視為指令加以執行，並將單引號字串視為文字字串指令，而您則可以使用雙引號將 *filenameset* 中括號檔案名稱括起。

變數取代

此外還加強了 for 變數參照的取代輔助按鍵。您可以選擇性使用下列語法 (對任何 I 變數)：

變數 (與輔助按鍵搭配)	描述
%~I	延伸會移除外圍括號 (? 的 %I
%~fI	將 %I 延伸為完整的路徑名稱
%~dI	只將 %I 延伸為磁碟機代號
%~pI	只將 %I 延伸為路徑
%~nI	只將 %I 延伸為檔案名稱
%~xI	只將 %I 延伸為副檔名
%~sI	將路徑延伸為只包含短檔名
%~aI	將 %I 延伸為檔案的檔案屬性
%~tI	將 %I 延伸為檔案的日期/時間
%~zI	將 %I 延伸為檔案的大小
%~$PATH:I	搜尋 PATH 環境變數中所列的目錄，並將 %I 延伸為找到之第一個之目錄的完整名稱。如果沒有定義環境變數名稱，或是搜尋找不到檔案，則此輔助按鍵便會延伸成空字串。

您可以組合這些輔助按鍵，取得複合式的結果：

變數 (搭配輔助按鍵)	描述
%~dpI	只將 %I 延伸為磁碟機代號及路徑
%~nxI	只將 %I 延伸為檔案名稱及副檔名
%~fsI	只將 %I 延伸為使用短檔名的完整路徑名稱
%~dp$PATH:I	在 PATH 環境變數所列的目錄中搜尋 %I，並將 %I 延伸為找到之第一個目錄的磁碟機代號及路徑名稱。
%~ftzaI	將 %I 延伸為類似 dir 的輸出行

附註

- 上述範例中的 *%I* 及 PATH，可以其他有效的值取代。%~ 語法則會被正確的 for 變數名稱所終止。
- 使用大寫的變數名稱 (如 *%I*) 可以提高程式碼的易讀性，並可避免和無需區分大小寫的輔助按鍵發生混淆。

附註

使用 in 及 do 關鍵字

In 及 do 不是參數，但卻是 for 指令中的必要項目。您如果省略了其中任一項關鍵字，Windows 2000 即會顯示錯誤訊息。

使用可取代的變數

若要避免與批次參數 *%0* 到 *%9* 發生混淆，請儘量在 *variable* 中使用 0 到 9 以外的字元。一般單純的批次程式可能只需使用單一字元的參數 (如 **%%f**) 便已足夠。

您可以在比較複雜的批次程式中採用多種數值作為 *variable* 的值，以區別各種可以取代的變數。

指定檔案群組

set 參數可以表示單一或數個檔案群組。您可以使用萬用字元 (* 及 ?) 指定檔案集合。下列是有效的檔案集合：

```
(*.doc)
(*.doc *.txt *.me)
(jan*.doc jan*.rpt feb*.doc feb*.rpt)
(ar??1991.* ap??1991.*)
```

當使用 for 指令時，*set* 的第一個值會取代 **%%***variable* (或 **%***variable*)，而 Windows 2000 則會使用指定的指令處理此值。在 Windows 2000 尚未處理完所有與 *set* 值對應的檔案 (或檔案群組) 之前，此作業會一直繼續。

範例

顯示目錄中的檔案

假設您要使用 **type** 指令顯示目前目錄中，所有檔案副檔名為 .doc 與 .txt 的內容。若要執行此作業，並使用可取代的變數 *%f*，請在命令提示字元後鍵入下列指令：

```
for %f in (*.doc *.txt) do type %f
```

在此範例中，目前目錄下所有副檔名為 .doc 或 .txt 的檔案，皆會取代 *%f* 變數，直到各檔案內容盡已顯示為止。若要在批次檔中使用此指令，只需以 *%%f* 取代所指定的各 *%f* 即可。否則，Windows 2000 會略過這些變數，而顯示錯誤訊息。

將輸出重新導向印表機

Windows 2000 可以讓您使用想要與指定指令搭乘使用的指令參數、管線及重新導向。例如，若要將上一個範例的輸出重新導向 PRN (預設的印表機連接埠)，請鍵入下列指令：

```
for %f in (*.doc *.txt) do type %f > prn:
```

剖析檔案

若要剖析檔案，但要略過註解行，可使用：

```
for /F "eol=; tokens=2,3* delims=," %i in (myfile.txt) do @echo %i %j %k
```

若要剖析 myfile.txt 的各行，但要略過以分號開頭的各行，再將每一行的第二及第三個 Token Ring 傳遞給 for 的主體 (Token Ring 須以逗號及 (或) 空格分隔)。請注意！**FOR** 敘述式主體會先參照 *%i*，取得第二個 Token Ring，並接著參照 *%j*，取得第三個 Token Ring，最後再參照 *%k*，取得第三個之後所剩餘的所有 Token Ring。對含有空格的檔名，您必須使用雙引號將該檔名括起。要在此使用雙引號，則必須使用 usebackq 選項；否則雙引號將會被解譯成需要剖析的文字字串。

%i 必須明確地在 FOR 敘述式中宣告，而 *%j* 及 *%k*，則可使用 tokens= 選項進行宣告。使用 tokens= 行時，只要指定的 Token Ring 不會嘗試宣告字母 z 或 Z 以上的變數，最多將可指定 26 個 Token Ring。

切記！for 變數名稱除需區分大小寫、通用性外，還不能夠在同一時間內啟用 52 個以上的此變數。

剖析字串

您也可利用將 *filenameset* 以單引號括起，並置於括弧內的方式，使用 for /F 剖析立即字串中的邏輯。其將會被視成檔案所提供的單行輸入而加以剖析。

剖析輸出

最後，您可以使用 for /F 指令，剖析指令的輸出。若要執行此作業，可將 *filenameset* 以反括號括起，並將其置於括弧。其將會被視成指令行而傳往子 Cmd.exe；輸出則會被擷取到記憶體中，並以檔案的方式進行剖析。見下列範例：

```
for /F "usebackq delims==" %i IN (`set`) DO @echo %i
```

此會列舉目前環境中的環境變數名稱。

Goto

將 Windows 2000 導向指定標籤所標記的批次程式中之位置。

goto 指令將 Windows 2000 導向批次程式中由標籤所識別的位置。Windows 2000 找到標籤時，其會執行從下一行開始的指令。

goto *label*

參數

label

指定 Windows 2000 要到達批次程式的位置。如果您啟用指令延伸 (Windows 2000 中的預設設定)，**goto** 指令的變更如下：

使用具有目標標籤 **:EOF** 的 **goto** 指令將控制傳輸到目前批次指令檔的尾端，可不必定義標籤，便可離開批次指令檔。使用具有 **:EOF** 標籤的 **goto** 指令時「必須」在標籤前插入冒號，例如：

```
goto :EOF
```

附註

標籤的有效值

label 參數可以包含空格，但不能包含其他分隔字元 (如分號或等號)。使用具有 EOF 標籤的 **goto** 指令時，「必須」在標籤前插入冒號，例如：

```
goto :EOF
```

Goto 使用每個標籤的前 8 個字元

goto 指令僅使用標籤的前 8 個字元。因此，標籤 :hithere01 及 :hithere02 都等同於 :hithere0。

用批次程式中的標籤對應標籤參數

在 goto 指令行中指定的標籤值必須對應批次程式中的標籤。批次程式中的標籤必須以冒號開始。

如果批次程式中不含指定的標籤，它將停止執行，且 Windows 2000 會顯示下列訊息：

```
Label not found
```

Windows 2000 認為以冒號 (:) 開始的批次程式行為標籤，而不將它當成指令來處理。如果是以冒號開始的位置，Windows 2000 將略過其上的任何指令。

在條件操作中使用 goto

範例

下列批次程式將磁碟機 A 中的磁片格式化為系統磁片。如果操作成功，goto 指令會將 Windows 2000 導向名為 end 的標籤位置。

```
echo off
format a:/s
if not errorlevel 1 goto end
echo An error occurred during formatting.
:end
echo End of batch program.
```

Pause

暫停批次程式的處理，並顯示提示使用者按任意鍵繼續的訊息。

pause

附註

提示使用者繼續程式

Windows 2000 在回應 **pause** 指令時會顯示下列訊息：

請按任意鍵繼續 . .

將批次檔劃分成區段

如果按 CTRL+C 來停止批次程式，則 Windows 2000 會顯示下列訊息：

要終止批次作業嗎 (Y/N)?

回應此訊息時，如果按 Y (是)，則批次程式會終止，並將控制傳回給作業系統。因此，可在不要處理的批次檔區段之前插入 **pause** 指令。**pause** 暫停批次程式的處理時，可按 CTRL+C，再按 Y，以停止批次程式

範例

假設想要批次程式提示使用者變更磁碟機中的磁片。若要做這樣的處理，可能需建立下列檔案：

```
@echo off
:begin
copy a:*.*
echo Please put a new disk into drive A
pause
goto begin
```

在此例中，會將磁碟機 A 之磁片上的所有檔案複製到目前的目錄。在顯示將其他磁片放入磁碟機 A 的註解提示後，pause 指令會暫停處理，以讓您變更磁片並按任意鍵繼續。此特殊的批次程式會以無窮迴圈來執行。goto BEGIN 指令會將指令直譯器傳送到批次檔的 begin 標籤。若要停止此批次程式，請按 CTRL+C，再按 Y。

Shift

變更批次檔中的可取代參數的位置。

shift

啟用指令副檔名時 (Windows 2000 的預設設定)，**shift** 指令可支援 **/n** 參數，這樣可讓指令在第 *n* 個引數時啟動轉換，而 *n* 值可以從 0 到 8。例如：

SHIFT /2

會將 **%3** 轉換為 **%2**、**%4** 轉換為 **%3** (依此類推)，而 **%0** 及 **%1** 不會受到影響。

附註

shift 指令的運作方式

shift 指令可將每個參數複製到上一個參數中，以變更可取代 *%0* 到 *%9* 參數的值。換言之，*%1* 的值會複製到 *%0*，而 *%2* 的值會複製到 *%1*，依此類推。這對撰寫可根據任意參數個數以執行相同操作的批次檔而言，十分有用。

使用 10 個以上的指令行參數

您也可使用 **shift** 指令，以建立可接受 10 個以上參數的批次檔。如果在指令行上指定 10 個以上的參數，則第十個參數後 (*%9*) 的參數會一次都轉換為 *%9*。

將參數轉換回去

沒有反向的 **shift** 指令。執行 **shift** 指令之後，就無法修復轉換之前所存在的第一個參數 (*%0*)。

範例

下列 Mycopy.bat 批次檔會顯示如何使用具有任意參數個數的 **shift** 指令。它會將檔案清單複製到指定的目錄。參數就是後面跟著任意個檔案名稱的目錄名稱。

```
@echo off
rem MYCOPY.BAT copies any number of files
rem to a directory.
rem The command uses the following syntax:
rem mycopy dir file1 file2 ...
set todir=%1
:getfile
shift
if "%1"=="" goto end
copy %1 %todir%
goto getfile
:end
set todir=
echo All done
```

Echo

打開或關閉指令回應功能，或顯示訊息。

echo [on | off] [*message*]

參數

on | off

指定是否打開或關閉指令回應功能。若要顯示目前的回應設定，請使用不帶任何參數的 **echo** 指令。

message

指定您要 Windows 2000 在螢幕上顯示的文字。

附註

使用帶有 echo 指令的訊息

當 echo 關閉時，**echo** *message* 指令是很有用的。若要顯示好幾行的訊息而不顯示其他指令，您可以在批次程式中的 **echo off** 指令後包括數個 **echo** *message* 指令。

隱藏命令提示字元

如果您在指令行上使用 **echo off** 指令，則命令提示字元就不會出現在螢幕上。若要重新顯示命令提示字元，請鍵入 **echo on**。

防止 Windows 2000 回應行

您可以在批次程式指令之前插入 at 符號 (@)，以防止 Windows 2000 回應該行。

回應空白行

若要在螢幕上回應空白行，您可以鍵入 **echo**，然後接著句點 (**echo.**)。其間必須沒有空格。

顯示管道及重新導向字元

若要在使用 **echo** 指令時顯示管道 (|) 或重新導向字元 (< 或 >)，請在管道或重新導向字元之前鍵入 ^ 字元 (例如，^>、^< 或 ^|)。如果您必須使用 ^ 字元本身 (^)，請一次使用二個 (^^)。

範例

下列範例顯示包括三行訊息、並跟著一個空白行的批次程式。

```
echo off
echo.
echo This batch program
echo formats and checks
echo new disks
echo.
```

如果您要關閉 echo 並不回應 **echo** 指令本身，請在指令之前包含一個 at 符號 (@)，如下所述：

```
@echo off
```

您可以在同一指令行上使用 **if** 及 **echo** 指令，如下所述：

```
if exist *.rpt echo The report has arrived
```

Setlocal

開始翻譯批次檔中的環境變數。除非遇到符合的 **endlocal** 指令，或到達批次檔結尾，否則會持續翻譯。

setlocal *option*

參數

option

啟用指令擴充時 (Windows 2000 的預設值)，**setlocal** 批次指令會接受選用的引數：*enableextensions* 或 *disableextensions*。這樣可啟用或停用指令擴充直到遇到符合的 **endlocal** 指令 (不管它們在 **setlocal** 指令之前的設定)。

將引數傳送給 **setlocal** 指令時，也會設定 *errorlevel* 值。如果給定兩個有效引數中的其中一個，則會將 *errorlevel* 值設定為零 (0)，否則，設定為一 (1)。

附註

使用 **setlocal** 指令，以在批次檔執行期間變更環境變數。在發出 **setlocal** 之後所做的環境變更，對批次檔而言是在本機上。遇到 **endlocal** 指令或批次檔終止時，會還原成先前的設定。

在批次程式 (巢狀) 中，可以有數個 **setlocal** 或 **endlocal** 指令。

範例

您可翻譯批次檔中的環境變數 (如下所示)：

```
  rem *******Begin Comment**************
rem This program starts the superapp batch program on the network,
rem directs the output to a file, and displays the file
rem in Notepad.
rem *******End Comment**************
```

```
@echo off
setlocal
path=g:\programs\superapp;%path%
call superapp>c:\superapp.out
endlocal
start notepad c:\superapp.out
```

在批次檔中測試指令擴充

如果給予兩個有效引數中的其中一個，則 **setlocal** 指令會將 *errorlevel* 值設定為零 (0)，否則設定為一 (1)。您可在批次指令檔中使用此指令，以判定擴充是否可用 (如下所示)：

```
    verify other 2>nul
setlocal enableextensions
if errorlevel 1 echo Unable to enable extensions
```

因為停用指令擴充或使用較舊版本的 Cmd.exe 時，**cmd** 無法設定 *errorlevel* 值，所以含無效引數的 **verify** 指令會將 *errorlevel* 值初始化為非零的值。同時，如果與有效擴充搭配使用的 **setlocal** 指令未將 *errorlevel* 值設定為一，則指令擴充不適用。

Endlocal

中止在批次檔案中的環境變更，將環境變數還原為相對應 setlocal 指令之前的原始值。

批次檔尾端有隱含的 endlocal 指令。

endlocal

如果已啟用指令擴充 (Windows 2000 預設)，則 endlocal 指令會將指令擴充還原為執行相對應的 endlocal 指令之前的已啟用或已停用狀態。

範例

您可以在批次檔中轉譯環境變數。

```
@echo off

rem This program starts the superapp batch program on the network,

rem directs the output to a file, and displays the file

rem in Notepad.

setlocal

path=g:\programs\superapp;%path%

call superapp>c:\superapp.out

endlocal

start notepad c:\superapp.out
```

Call

從其他批次程式呼叫批次程式，而不會造成父系批次程式停止。call 指令已可接受標籤作為呼叫目標。

call [*drive*:][*path*] *filename* [*batch-parameters*]

call *:label* [*arguments*]

參數

[*drive*:][*path*] *filename*

指定所要呼叫之批次程式的位置及名稱。*filename* 參數的副檔名必須是 .bat 或 .cmd。

批次參數

指定批次程式所需的指令行資訊。請參閱 *arguments* 參數中 *batch-parameters* 的相關資訊。

:label

指定批次程式控制跳往的標籤。您可以將 call 指令與此參數搭配使用，建立新的批次檔內容，並將控制傳遞給指定標籤之後的敘述式。在第一次執行到批次檔的尾端 (跳到標籤之後) 時，控制會傳回 CALL 敘述式之後的敘述式。而在第二次執行到批次檔的尾端時，批次指令檔便會結束。

附註

使用批次參數

批次參數內可以含有任何您想要傳給批次程式的資訊，包括開關、檔名、可替換參數 (*%1* 到 *%9*) 及變數 (如 *%baud%*)。

使用管道及重新導向符號

請勿將 call 指令與管道及重新導向符號搭配使用。

範例

若要從其他批次程式執行 Checknew.bat 程式，請將下列指令納入父系批次程式中：

```
call checknew
```

假設父系批次程式可接受兩個可替換參數，且您希望該程式能夠將這兩項參數傳給 Checknew.bat，您可以在父系批次程式中使用下列指令：

```
call checknew %1 %2
```