# 

**Article:** Multi-line IF statements in batch file

**Date**: 17/02/2012

**Posted by**: HeelpBook Staff

**Source**: [Link](http://www.cynosurex.com/Forums/DisplayComments.php?file=Batch/Multi-Line_IF_Statements_in_Batch_File)

**Permalink**: [Link](http://heelpbook.altervista.org/2012/dos-multi-line-if-statements-in-batch-file/)

# [**DOS – Multi-Line IF Statements in Batch File**](http://heelpbook.altervista.org/2012/dos-multi-line-if-statements-in-batch-file/)

Most folks don't realize it, but you can actually place multiple statements into if-else clauses. In fact, most people don't even realize that the [gs batch] file allows you to have an else clause. The following code demonstrates how you can have multi-line **IF** statements, so that you don't have to do a lot of goto's.

The **'('** and **')'** must be on the same line as the **if** and **else** statements as shown above. Even though they are not the open and close bracket, you can think of them as such. With **multi-line** if-else statements, you batch file can become far more structured than ever.

**What doesn’t work and How to fix it**

The set variable function does not work inside a multi-statement if-else scope. For example, the following code does not work:

if "%tempDir%" == "" (

set output=\*\*\* Variable tempDir not set!

echo %output%

echo %output% >> %logFile%

set success=false

)

The batch processor was never able to use the **%output%** that was set within that scope.

**SOLUTION 1**

The workaround for this problem is to call a **sub-routine** in the if statement. The following is an example:

if "%tempDir%" == "" (

call :subroutine

)

goto :end

:subroutine

set output=\*\*\* Variable tempDir not set!

echo %output%

echo %output% >> %logFile%

set success=false

goto :eof

:end

**SOLUTION 2**

SetLocal EnableDelayedExpansion

if "" == "" (

set output=\*\*\* hello dude

echo !output!

)

echo !output!

echo %output%

EndLocal