Batch file and DEL errorlevel 0 issue

stackoverflow.com/questions/22953027/batch-file-and-del-errorlevel-0-issue

4

The batch has to remove files and directories from specific locations and output success or stdout/stderr messages to a new .txt file. I have created the most of the script and it performs exactly as it should, except when the deletion is successful it moves forward to the next line rather than echo a 'successful' message on the log.

```
echo Basic Deletion Batch Script > results.txt
@echo off
call :filelog >> results.txt 2>&1
notepad results.txt
exit /b

:filelog

call :delete new.txt
call :delete newer.txt
call :delete newest.txt
call :remove c:\NoSuchDirectory

GOTO :EOF

:delete
echo deleting %1
del /f /q c:\Users\newuser\Desktop\%1
if errorlevel 0 echo succesful

GOTO :EOF

:remove
echo deleting directory %1
rmdir /q /s %1

GOTO :EOF
```

For some reason I can't find the syntax for if del succeeds echo 'successful'. In the above example if I remove the line

```
if errorlevel 0 echo successful
```

Everything works fine, but no success message. With this line left in it echoes success for every line.

edited Dec 5 '19 at 20:33

aschipfl
28.2k1010 gold badges4444 silver badges7575 bronze badges

asked Apr 9 '14 at 4:50

John Scott
4111 gold badge11 silver badge44 bronze badges

Add a comment

7 Answers

Active Oldest Votes
14

`del` and `ErrorLevel`?

The `del` command does not set the `ErrorLevel` as long as the given arguments are valid, it even resets the `ErrorLevel` to `0` in such cases (at least for Windows 7).

`del` modifies the `ErrorLevel` only in case an invalid switch is provided (`del /X` sets `ErrorLevel` to `1`), no arguments are specified at all (`del` sets `ErrorLevel` to `1` too), or an incorrect file path is given (`del :` sets `ErrorLevel` to `123`), at least for Windows 7.

Possible Work-Around

A possible work-around is to capture the `STDERR` output of `del`, because in case of deletion errors, the related messages (`Could Not Find [...]`, `Access is denied.`, `The process cannot access the file because it is being used by another process.`) are written there. Such might look like:

```
for /F "tokens=*" %%# in ('del /F /Q "\path\to\the\file_s.txt" 2^>^&1 1^> nul') do
(2> nul set =)
```

To use the code in command prompt directly rather than in a batch file, write `%#` instead of `%%#`.

If you do not want to delete read-only files, remove `/F` from the `del` command line;
if you do want prompts (in case wildcards `?` and/or `*` are present in the file path), remove `/Q`.

Explanation of Code

This executes the command line `del /F /Q "\path\to\the\file_s.txt"`. By the part `2>&1 1> nul`, the command output at `STDOUT` will be dismissed, and its `STDERR` output will be redirected so that `for /F` receives it.

If the deletion was successful, `del` does not generate a `STDERR` output, hence the `for /F` loop does not iterate, because there is nothing to parse. Notice that `ErrorLevel` will not be reset in that case, its value remains unchanged.

If `for /F` recieves any `STDERR` output from the `del` command line, the command in the loop body is executed, which is `set =`; this is an invalid syntax, therefore `set` sets the `ErrorLevel` to `1`. The `2> nul` portion avoids the message `The syntax of the command is incorrect.` to be displayed.

To set the `ErrorLevel` explicitly you could also use `cmd /C exit /B 1`. Perhaps this line is more legible. For sure it is more flexible because you can state any (signed 32-bit) number, including `0` to clear it (omitting the number clears it as well). It might be a bit worse in terms of performance though.

Application Example

The following batch file demonstrates how the above described work-around could be applied:

```
:DELETE
echo Deleting "%~1"...
rem this line resets ErrorLevel initially:
cmd /C exit /B
rem this line constitutes the work-around:
for /F "tokens=*" %%# in ('del /F /Q "C:\Users\newuser\Desktop\%~1" 2^>^&1 1^> nul')
do (2> nul set =)
rem this is the corrected ErrorLevel query:
if not ErrorLevel 1 echo Deleted "%~1" succesfully.
goto :EOF
```

Presetting `ErrorLevel`

Besides the above mentioned command `cmd /C exit /B`, you can also use `> nul ver` to reset the `ErrorLevel`. This can be combined with the `for /F` loop work-around like this:

```
> nul ver & for /F "tokens=*" %%# in ('del /F /Q "\path\to\the\file_s.txt" 2^>^&1 1^>
nul') do (2> nul set =)
```

Alternative Method Without `for /F`

Instead of using `for /F` to capture the `STDERR` output of `del`, the `find` command could also be used like `find /V ""`, which returns an `ErrorLevel` of `1` if an empty string comes in and `0` otherwise:

```
del "\path\to\the\file_s.ext" 2>&1 1> nul | find /V "" 1> nul 2>&1
```

However, this would return an `ErrorLevel` of `1` in case the deletion has been successful and `0` if not. To reverse that behaviour, an `if`/`else` clause could be appended like this:

```
del "\path\to\the\file_s.ext" 2>&1 1> nul | find /V "" 1> nul 2>&1 & if ErrorLevel 1
(1> nul ver) else (2> nul set =)
```

Different Approach: Checking File for Existance After `del`

A completely different approach is to check the file for existence after having tried to delete it (thanks to user Sasha for the hint!), like this, for example:

```
del /F /Q "\path\to\the\file_s.txt" 1> nul 2>&1
if exist "\path\to\the\file_s.txt" (2> nul set =) else (1> nul ver)
```

edited Sep 9 '20 at 9:09

answered Oct 28 '15 at 23:41

aschipfl
28.2k1010 gold badges4444 silver badges7575 bronze badges

> Sure, @Sasha, that is of course also possible; you could write your own answer to show how it could be done, if you want; or do you prefer me to extend mine instead? – aschipfl Jul 11 '19 at 9:27

Add a comment
1

When using this syntax, instead of this

```
if errorlevel 0 echo successful
```

you can use this - because errorlevel 0 is always true.

```
if not errorlevel 1 echo successful
```

answered Apr 9 '14 at 5:48

foxidrive
37.2k88 gold badges4646 silver badges6666 bronze badges

Add a comment
1

Just use `rm` from UnxUtils (or gow or cygwin). It sets the errorlevel correctly in case of a nonexistent file, or any errors deleting the file.

edited Apr 8 '19 at 19:19

Bob Stein
11.8k88 gold badges6868 silver badges8989 bronze badges

answered Oct 15 '18 at 22:46

clueless
2122 bronze badges

Add a comment
0

This was added as an edit by the original asker, I have converted it to a community wiki answer because it should be an answer, not an edit.

I found out how to do it... one way anyway.

```
echo Startup > results.txt
@echo off
call :filelog >> results.txt 2>&1
notepad results.txt
exit /b

:filelog

call :delete new.txt
call :delete newer.txt
call :delete newest.txt
call :remove c:\NoSuchDirectory

GOTO :EOF

:delete
echo deleting %1
dir c:\users\newuser\Desktop\%1  >NUL 2>&1
SET existed=%ERRORLEVEL%
del /f /q c:\Users\newuser\Desktop\%1
dir c:\users\newuser\Desktop\%1 2>NUL >NUL
if %existed% == 0 (if %ERRORLEVEL% == 1  echo "successful" )

GOTO :EOF

:remove
echo deleting directory %1
rmdir /q /s %1

GOTO :EOF
```

answered Jun 24 '15 at 16:30

community wiki

durron597
Add a comment
0

IF ERRORLEVEL 0 [cmd] will execute every time because IF ERRORLEVEL # checks to see if the value of ERRORLEVEL is greater than or equal to #. Therefore, every error code will cause execution of [cmd].

A great reference for this is: http://www.robvanderwoude.com/errorlevel.php

```
>IF /?
Performs conditional processing in batch programs.

IF [NOT] ERRORLEVEL number command
IF [NOT] string1==string2 command
IF [NOT] EXIST filename command

  NOT               Specifies that Windows should carry out
                    the command only if the condition is false.

  ERRORLEVEL number Specifies a true condition if the last program run
                    returned an exit code equal to or greater than the number
                    specified.
```

I would recommend modifying your code to something like the following:

```
:delete
echo deleting %1
del /f /q c:\Users\newuser\Desktop\%1
if errorlevel 1 (
    rem This block executes if ERRORLEVEL is a non-zero
    echo failed
) else (
    echo succesful
)

GOTO :EOF
```

If you need something that processes more than one ERRORLEVEL, you could do something like this:

```
:delete
echo deleting %1
del /f /q c:\Users\newuser\Desktop\%1
if errorlevel 3 echo Cannot find path& GOTO :delete_errorcheck_done
if errorlevel 2 echo Cannot find file& GOTO :delete_errorcheck_done
if errorlevel 1 echo Unknown error& GOTO :delete_errorcheck_done
echo succesful
:delete_errorcheck_done

GOTO :EOF
```

OR

```
:delete
echo deleting %1
del /f /q c:\Users\newuser\Desktop\%1
goto :delete_error%ERRORLEVEL% || goto :delete_errorOTHER

:delete_errorOTHER
echo Unknown error: %ERRORLEVEL%
GOTO :delete_errorcheck_done
:delete_error3
echo Cannot find path
GOTO :delete_errorcheck_done
:delete_error2
echo Cannot find file
GOTO :delete_errorcheck_done
:delete_error0
echo succesful
:delete_errorcheck_done

GOTO :EOF
```

edited Sep 22 '15 at 1:43

answered Sep 22 '15 at 1:38

Hossy
12511 silver badge88 bronze badges

Add a comment
0

The answer of **aschipfl** is great (thanks, helped me a lot!) using the code under *Presetting ErrorLevel* you get a nice standard function:

Take care to use `%~1` instead of `%1` in the `del` statement, or you will get errors if you use a quoted filename.

```
::#######################################################################
::call :DELETE "file.txt"
::call :DELETE "file.txt" "error message"
:DELETE
  >nul ver && for /F "tokens=*" %%# in ('del /F /Q "%~1" 2^>^&1 1^> nul') do (2>nul
set =) || (
    if NOT .%2==. echo %~2
  )
goto :EOF
```

BTW 1: You can give a nifty error message as a second parameter
BTW 2: Using `::` instead of `REM` for comments makes the code even more readable.

answered Mar 30 '17 at 16:29

CONSULitAS
122 bronze badges

Add a comment
0

Code:

Error Code: *(What you did)*

```
if errorlevel 0 echo succesful
```

The problem here is that you aren't calling errorlevel as a variable and plus you didn't add in the operator to the statement as well.

Correct Code: *(Here is what it should actually be.)*

```
if %ERRORLEVEL% EQU 0 echo succesful
```

Definitions:

EQU: The EQU stands for Equal. This kind of operator is also called a relational operator. Here is the documentation link to operators if you wanna know more, there are other ones but this helped me.

ERRORLEVEL: is declared as a variable and usually get the error level of the last command run usually. Variables are usually called when they are between percent signs like this

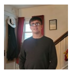
```
%foo%
```

For some more help on variables, go to cmd (Which you can go to by searching it on windows 10) and type in "set /?", without the quotes. the set command is the command you use to set variables

edited Jul 3 '20 at 8:27

halfer
18.5k1212 gold badges7777 silver badges157157 bronze badges

answered Jun 21 '20 at 16:19



Jonathan J. Pecany
10788 bronze badges

Add a comment

Your Answer

Not the answer you're looking for? Browse other questions tagged batch-file cmd echo delete-file errorlevel or ask your own question.