Windows Batch Scripting: Advanced Tricks

- Overview
- Part 1 – Getting Started
- Part 2 – Variables
- Part 3 – Return Codes
- Part 4 – stdin, stdout, stderr
- Part 5 – If/Then Conditionals
- Part 6 – Loops
- Part 7 – Functions
- Part 8 – Parsing Input
- Part 9 – Logging
- Part 10 – Advanced Tricks

Boilplate info

I like to include a header on all of scripts that documents the who/what/when/why/how. You can use the `::` comment trick to make this header info more readable:

```
:: Name:     MyScript.cmd
:: Purpose:  Configures the FooBar engine to run from a source control tree path
:: Author:   stevejansen_github@icloud.com
:: Revision: March 2013 - initial version
::           April 2013 - added support for FooBar v2 switches

@ECHO OFF
SETLOCAL ENABLEEXTENSIONS ENABLEDELAYEDEXPANSION

:: variables
SET me=%~n0


:END
ENDLOCAL
ECHO ON
@EXIT /B 0
```

Conditional commands based on success/failure

The conditional operators `||` and `&&` provide a convenient shorthand method to run a 2nd command based on the succes or failure of a 1st command.

The `&&` syntax is the AND opeartor to invoke a 2nd command when the first command returns a zero (success) exit code.

```
DIR myfile.txt >NUL 2>&1 && TYPE myfile.txt
```

The `||` syntax is an OR operator to invoke a 2nd command when the first command returns a non-zero (failure) exit code.

```
DIR myfile.txt >NUL 2>&1 || CALL :WARNING file not found - myfile.txt
```

We can even combined the techniques. Notice how we use the `()` grouping construct with `&&` to run 2 commands together should the 1st fail.

```
DIR myfile.txt >NUL 2>&1 || (ECHO %me%: WARNING - file not found - myfile.txt >2
&& EXIT /B 1)
```

Getting the full path to the parent directory of the script

```
:: variables
PUSHD "%~dp0" >NUL && SET root=%CD% && POPD >NUL
```

Making a script sleep for N seconds

You can use `PING.EXE` to fake a real *nix style `sleep` command.

```
:: sleep for 2 seconds
PING.EXE -N 2 127.0.0.1 > NUL
```

Supporting “double-click” execution (aka invoking from Windows Explorer)

Test if `%CMDCMDLINE%` is equal to `%COMSPEC%` If they are equal, we can assume that we are running in an interactive session. If not equal, we can inject a PAUSE into the end of the script to show the output. You may also want to change to a valid working directory.

```
@ECHO OFF
SET interactive=0

ECHO %CMDCMDLINE% | FINDSTR /L %COMSPEC% >NUL 2>&1
IF %ERRORLEVEL% == 0 SET interactive=1

ECHO do work

IF "%interactive%"=="0" PAUSE
EXIT /B 0
```