Windows Batch Scripting: Logging

I use basic logging facilities in my scripts to support troubleshooting both during execution and after execution. I use basic logging as a way to instrument what my scripts are doing at runtime and why. I remember watching a network operations center trying to troubleshoot a legacy batch process where the sysadmins literrally had to try to read the lines of a console window as they trickled by. This technique worked fine for years when the batch machines used dial-up modems for connectivity to remote resources. However, the advent of brooadband meant the batch script executed faster than anyone could read the output. A simple log file would have made troubleshooting work much easier for these sysadmins.

Log function

I really like the basic `tee` implementation I wrote in Part 7 – Functions.

```
@ECHO OFF
SETLOCAL ENABLEEXTENSIONS

:: script global variables
SET me=%~n0
SET log=%TEMP%\%me%.txt

:: The "main" logic of the script
IF EXIST "%log%" DELETE /Q %log% >NUL

:: do something cool, then log it
CALL :tee "%me%: Hello, world!"

:: force execution to quit at the end of the "main" logic
EXIT /B %ERRORLEVEL%

:: a function to write to a log file and write to stdout
:tee
```

```
ECHO %* >> "%log%"
ECHO %*
EXIT /B 0
```

This `tee` quasi function enable me to write output to the console as well as a log file. Here I am reusing the same log file path, which is saved in the users `%TEMP%` folder as the name of the batch file with a .txt file extension.

If you need to retain logs for each execution, you could simply parse the %DATE% and %TIME% variables (with the help of command line extensions) to generate a unique filename (or at least unique within 1-second resolution).

```
REM create a log file named [script].YYYYMMDDHHMMSS.txt
SET
log=%TEMP%\%me%.%DATE:~10,4%_%DATE:~4,2%_%DATE:~7,2%%TIME:~0,2%_%TIME:~3,2%_%TIME:
~6,2%.txt
```

Taking a queue from the *nix world, I also like to include a prefix custom output from my own script as `script: some message`. This technique drastically helps to sort who is complaining in the case of an error.

Displaying startup parameters

I also like to display the various runtime conditions for non-interactive scripts, like something that will be run on a build server and redirected to a the build log.

Sadly, I don’t know of any DOS tricks (yet) to discrimintate non-interactive sessions from interactive sessions. C# and .Net has the `System.Environment.UserInteractive` property to detect if this situation; *nix has some tricks with tty file descriptors. You could probably hack up a solution by inspecting a custom environmental variable like `%MYSCRIPT_DEBUG%` that defaults to being false.