þÿBATCH FILE COMMANDS
Simple programming commands in a batch environment

Yeah, yeah, I know that many people think batch files are mostly things of the past. Sometimes, though, a well-conceived batch file is just the thing to automate the job you want to do.

I am not going to cover all the theory and practice of batch files from the ground up. Any good book on DOS (now found in the Antiquities section of your local library <g>), and many of the best on Windows, will have a section on batch files. Simply put, a batch file is a plaintext file with a name ending in .BAT. In its simplest form, it contains a series of commands that could be executed from a command prompt (system prompt). The batch file simply autoexecutes them for you. (In fact, AUTOEXEC.BAT is the best known, and most widely used, batch file.) To execute a batch file, type its name at a command prompt, or execute a Windows shortcut that does the same thing.

The simplest idea of how to write a batch file is: Figure out how you would type the commands at a DOS prompt, then type them, one per line, in a text file   and you ve written your batch file.

However, there are also more sophisticated batch file structures, using simple programming commands built into the batch structure. This article summarizes the most important of these.

Commandline Arguements {%x}

Variables can be inserted into a batch structure in the form of command line arguements. These are in the form %1, %2, etc. To populate the variables, type the desired values after the batch file name when executing it.

DOS Environment Variable Names {%nn%}

DOS environment variables also can be used as variables in batch files. For example:

COPY %windir%\filename a:

Where does one get a list of DOS environment variables? I have never found a comprehensive list; but a partial but lengthy list of existing environment variables can be gotten by typing SET at a command prompt.

And here s the really cool part! You can make them up as you go along, and assign them as you wish (as long as you don t grab one that has a legitimate assigned value, such as, say, %windir%, the Windows directory name!). Pick a name, populate it with the SET command by any means known to you (including having one batch file run a process that includes setting one, and then another batch file using it), then use it by placing the name between flanking % signs. Environment variables remain until overwritten, or until a reboot. (If you set them in a DOS window, they will end when that session is closed.)

If you precede an environment variable setting with the SETLOCAL command (on a line of its own), then environment variable changes are local to the batch file. They do not exist for any other process and they do not survive the completion of the batch file s execution. You can turn this setting off by issuing an ENDLOCAL command later in the batch file.

Silly example: To change the current logged drive to D:, do the following:

SET GONEXT=D:
%GONEXT%

More practical example: You want to copy a file to the desktop of each user the next time they log into Windows. Each user logs into a different user profile, and the Desktop folder is in a unique location for each user. (The folder name will, of course, vary on non-English versions of Windows.) For a file called MYFILE.TXT, you can do this as follows on Windows 2000 or XP computers by using an environment variable %userprofile% which gives the path to the root of a given user s profile:

COPY MYFILE.TXT %userprofile%\Desktop

START Command

The START command can launch a Windows program either by specifying the program name (and its command-line parameters), or by specifying a data file name that is associated with a particular program (one that would automatically launch if you clicked on it in Windows).

For example, if you have NOTEPAD.EXE associated with all TXT files, then you could open the file SOME.TXT in any of the following four ways:

NOTEPAD SOME.TXT
SOME.TXT
START NOTEPAD.EXE SOME.TXT
START SOME.TXT

Why use one or the other? Well, sometimes you may have to use one particular form to get a result depending, for example, on how the particular program is coded. Though the first form usually will work, you may want, for example, to write a more general batch file to open any particular program and associated file without knowing what the requirements of all such files might be. You could, then, write a general batch file line such as START %1% %2%.

One particular use of the START command is to launch the default browser and go directly to a URL, for example: START http://google.com

You may use any of four command line parameters with the START command. These go after the word START, but before the program name:

/minimized or /m
/maximized or /max
/restored or /r
/wait or /w

The first three determine the screen status in which the program opens. The last one forces the batch file to halt processing until the called program has finished executing. (This can be useful, for example, if you are loading multiple items in your windows Startup folder, and the nature of the programs require that one be finished before the next starts loading. Put them all in a single batch file, using the /wait parameter, and only put a shortcut to the batch file in the Startup folder.) Command line parameters of the START command can be combined in a single line. Example:

START /max /wait NOTEPAD.EXE SOME.TXT

IF and IF NOT Commands

There are three variations of the IF and IF NOT commands.

IF EXIST: Execute the commandline only if a particular file exists:

IF EXIST some.txt COPY c:\some.dll %windir%\SYSTEM\some.dll
Compare two text strings, and execute the commandline only if they are identical.

IF %HostWinBootDrv%==C SET WinDir=C:\WINDOWS
Error testing: Check the exit code of the most recently run program. If it is equal to or greater than the number specified, execute the command:

IF ERRORLEVEL 4 ERASE trashfile.tmp /P

GOTO Command

You can set a label in a batch file by beginning a line with a colon. You can then go directly to that label with the GOTO command. The GOTO command searches both forward and backward in the batch file; that is, it simply goes to the label location, regardless of where it is in the file.

For example, in my batch file for removing the Happy99 virus, UNHAPPY.BAT, the following code was used to make sure a file was not deleted unless the original form of it (backed up by the virus under the name WSOCK32.SKA) is present:

IF NOT EXIST WSOCK32.SKA GOTO SavedIt
DEL WSOCK32.DLL
RENAME WSOCK32.SKA WSOCK32.DLL
:SavedIt

FOR Command

The syntax for this command is: FOR variable in (set list) DO command

The variable must be in the form of one alphabetic character preceeded by %%; e.g., %%v.

NOTE: The %% is necessary because this is in a batch file which, otherwise, would give a special meaning to a single %. However, if you run the FOR command outside of a batch file, simply from the system prompt, just use a single % in the variable name. (Tip from Steve Wisdom)

The set list is enclosed within parentheses. These values will be assigned to the variable successively. You can use any text enclosed in quotes, batch file commandline parameters, environment variables, or DOS file wildcard expressions.

The command can be any valid command that otherwise could be entered into the batch file as a line of its own. example:

FOR %%D in (SYSTEM, COMMAND, SHELLNEW, "Start Menu") DO DIR "%windir%\%%D" /W

Menu Creation

Sometimes you may want to let a batch file branch one way or another based on user input. This is especially helpful when you have several related batch processes and would like to combine them into a single application.

Back in DOS days, a common approach was to construct menus with multiple batch files. For example, you could

create one batch file called MENU.BAT that displayed the menu (a series of text lines), inviting a user to type a 1, 2, 3, etc. (or A, B, C, etc.) to choose an option (a program to run, or archiving process, or whatever). That menu batch file would end, delivering the user back to a command prompt. You would then have a series of other batch files called 1.BAT, 2.BAT, 3.BAT, etc. so that, when the user typed (for example) 2 and pressed Enter, it would run 2.BAT. (This is way easier to understand if you walk through making one! It s really terribly simple.)

Today, when users don t live in a DOS command prompt world, we want something slightly more sophisticated and, fortunately, we have it. There is a pretty cool way to allow user input in Windows 2000 and XP, and even better ways that work in Windows Vista.

In Windows 2000 or XP, the SET command has new /A and /P flags that allow user input. The latter is especially helpful for our present purposes. You can accept user input and assign it to a system variable with the following code:

SET /P variable=PromptString

The PromptString is optional. If you include one, it will be displayed on the screen. (Don" t forget a space at the end of the prompt if you want one!) For example,

SET /P M=Type 1 or 2, then press ENTER.

will display on the monitor the phrase Type 1 or 2, then press ENTER. It will then wait for the user to type something and press Enter. It will then assign whatever character the user types to the system variable %M%, which you can use in other batch file commands.

Windows Vista has added the CHOICE command. This is pretty cool! It lets you build simple menus just from this one command. On a Windows Vista computer, open a command prompt and type CHOICE /? to see all the things you can do with it. At the present, this might not be so useful if yo uare writing batch files that also will be run on Windows XP computers, because the CHOICE command will not work on those computers   and the SET /P approach above still works in Vista.

Here is an example of a batch file I recently wrote for my office. It uses many of the features discussed on this page, including menu creation. The problem to be solved was that (for reasons too tedious for the present article) users accessing our network remotely no longer had access to their browser Favorites. Additionally, it was useful (when swapping out computers) to migrate a user s Favorites from the old computer to the new. Both of these could be solved by moving the Favorites (which are simply shortcut files) up onto a personal network drive (let s call it P:) to which they always had access. I wanted to allow the user, with a single file that I could email them, to be able both to cache their Favorites on the network drive and to pull these back down to another computer. Here is a slightly edited version of the batch file.

```
ECHO OFF
CLS
:MENU
ECHO.
ECHO .............................................
ECHO PRESS 1 or 2 to select your task, or 3 to EXIT.
ECHO .............................................
ECHO.
ECHO 1 - Export Favorites from COMPUTER to PERSONAL DRIVE (P:)
ECHO 2 - Import Favorites from PERSONAL DRIVE (P:) to COMPUTER
ECHO 3 - EXIT
ECHO.
SET /P M=Type 1, 2, or 3, then press ENTER:
```

```
IF %M%==1 GOTO EXPORT
IF %M%==2 GOTO IMPORT
IF %M%==3 GOTO EOF
:EXPORT
XCOPY "%userprofile%"\Favorites\*.* P:\Favorites\ /S/Y
GOTO MENU
:IMPORT
XCOPY P:\Favorites "%userprofile%"\Favorites\*.* /S
GOTO MENU
```

More Information on These Commands

Each of these options (START, IF, GOTO, FOR, SET) is an actual DOS command. At a system prompt, type the command name followed by /? to get further help on these items.

Note that there may be particular capabilities that show up in one version of Windows, but not in another. For example, though DOS per se may well be dead in Windows XP and Vista, the commandline functions people most often associate with DOS are not dead at all! (We just don t call them DOS commands anymore; we call them command prompt commands. But they re the same thing.) In some cases, these commands have been made more powerful in Windows XP. In particular, if Win XP Command Extensions are enabled, each of these four has very greatly enhanced capabilities (see Win XP Help & Support to learn about Command Extensions). Take advantage of the opportunity to explore each of them with the /? help flag.

   SOME EXAMPLES (that don t even require Command Extensions): START has new flags that let you set the priority (/LOW, /NORMAL, /HIGH, etc.) of the application you are launching. IF has an ELSE option than greatly expands its power, but which requires a somewhat complicated syntax sometimes (which the /? help text covers reasonably well).

Since these START, IF, GOTO, and FOR are actual OS commands, they can be used from a system prompt just like DIR, COPY, or any other DOS command. This means that they can be used outside of a batch file as well. There are small differences or issues that you can easily discover in use, and discussion of which would go beyond the purpose of the present page. For anyone comfortable working at a DOS system prompt, this should present no significant problem. Just remember:

 ///////////*1000+ HACKING TRICKS & TUTORIALS - ebook By Mukesh Bhardwaj Blogger - Paid Version - only @ TekGyd | itechhacks | Mukeshtricks4u*////////