Contents

Introduction

"Why use batch files?", you might ask as you read this. Well, there is one very easy and simple reason- time. Batch files will perform tasks that you don't want to do all the time. For example, one file that we overlook a lot is the AUTOEXEC.BAT. This file is the file that your computer accesses every time you turn it on. It is located in your root directory, usually C:\. Now, lets say you have some kind of menu in your program. It gives you a choice of Microsoft Windows, Microsoft Word, and Microsoft Excel. Now lets call this program MENUMAKER. This is probably how your AUTOEXEC.BAT would look if you had MENUMAKER in it after the basic autoexec commands...

```
CD\MENUMAK
MENUMAK C:\MENUMAK
CD\
```

All this basically means is that you have a lot of commands to type but your AUTOEXEC.BAT does them automatically.

Batch files also give you some experience in programming. I started with them when I was younger in programming and they really help you prepare for Basic or C programming. The If and Goto command are very similar in Basic and in Batch file processing. Also, batch processing gives you experience with MS-DOS. It makes it nice that everytime you learn a new MS-DOS Command, you can apply it to your batch file. Well, your on your way to making successful batch files with BatchEdit, good luck.

Contents

Batch File Commands

There are some basic batch file concepts and commands you must know to get started in your batch file processing. Our first will be the **echo** command and the manipulation of it. Echo has the following syntax: **Echo On** or **Echo Off**. Echo On specifies that a DOS Prompt will show as well as the command. Echo Off, which is preferred, and gives a more professional look to batch files, specifies no prompt or batch command will show. It provides a smoother, flowing look.
A third use of the echo command is the **echo** *message*. This displays a message you write. Perhaps you wanted a message "Hello World!". The first lines of your batch file should be:

```
ECHO OFF
ECHO HELLO WORLD!
```

For any reason, if you or the batch file user should want to interrupt their batch file, the **Ctrl-C** keys should be used. These interrupt your batch files if you have a loop or other bug.

Another command you should keep in mind if you want to use, for it has no real batch file value except your own, is the **REM** command. **REM** *message*is used when you want to display a message. This is particularly useful in BatchEdit because when you make your programs, it is likely that after a week of not seeing them, you'll forget what they were all about. It is also good if you have a batch file that you upload as an example to CompuServe, one where people will be really looking at it and studying it.

The next command we want to learn is the **Pause** command. Pause allows you to wait for the user to press a key. Use the pause command as a command by itself. It is used if you have made a batch file to make diskcopy or to format a disk. You don't want your file to format a blank disk so you use the pause command. Pause displays the message "Press any key to continue". If you don't want any message use the **PAUSE > NUL** command. This will display nothing, but it will pause the batch file until the user presses a key. Use the echo command with the pause command to display a message for the pause command. For example...

```
ECHO OFF
ECHO PRESS A KEY IF YOU WANNA KEEP GOING, PRESS CTRL-C TO STOP ME
PAUSE > NUL
```

That's an overview of basic batch file concepts. Memorize and apply these. If your a Visual Basic programmer it's like the Form1.Caption property or for QBasic programmers its CLS, which still applies in MS-DOS.

NoteTo learn more commands such as DISKCOPY, FORMAT, or DIR, consult your MS-DOS Reference.

Using Batch File Parameters in Your Programs

Batch File parameters are a great way to use and manipulate user input. A batch file parameter is something the user types when he or she types in the batch file parameters. For example, the batch file name might be MANAGE.BAT. When the user types MANAGE F B, the "F" and the "B" are the command-line parameters. They are assigned to %1 and %2. There is nine paramaters. The 0% parameter is the batch file name. The % variables are great for displaying and using. To display the %1 use this program:

```
ECHO OFF
CLS
ECHO %1
```

If you want to display "%1" use the Echo command with %%1 instead of %1. %1 displays the actual parameter itself. Actual commands like DISKCOPY use command-line parameters. DISKCOPY A: B: means diskcopy from the "A" drive to the "B" drive. Let's use command-line parameters to make an advanced CHKDSK program called CHKDSK1:

```
ECHO OFF
CLS
%1
ECHO RUNNING CHKDSK1 ON YOUR %1 DRIVE
CHKDSK %1
VOL
DIR /O /P
PAUSE > NUL
ECHO PRESS A KEY TO GO BACK TO BATCHEDIT
```

If your new to batch files, all this about parameters means nothing. However, once you learn to manipulate them, as in the above example, which does this to a small degree. In the next section you'll learn probably the "meat" of this help file.

Contents

The More "Advanced" Batch File Concepts

After you get finished reading this section, you'll learn that the IF operator is very important operator. Lets look at the first use of the IF, the **IF EXIST**. Use the IF EXIST to test if a file name exists. Let's look at a quick example of the **IF EXIST** operator:

```
ECHO OFF
CLS
IF EXIST AUTOEXEC.BAT ECHO THE AUTOEXEC.BAT FILE IS IN THIS DIRECTORY
TYPE AUTOEXEC.BAT | MORE
```

You might expect, the **IF NOT EXIST** operator is the exact opposite of the IF EXIST. It is used with the same syntax **IF NOT EXIST** *filename.ext DOSCommand*.

Another use of the IF property is the **IF** *StringOne***==** *StringTwo DOSCommand*
When we talk about strings, for those who don't program in Basic or C, which prolifically use strings. First, strings are enclosed in double quotation marks ("*String*"). They are enclosed in quotation marks because words not enclosed in quotation marks are variables. Batch files don't use variable much so don't be concerned with them yet. Let's try and IF command with strings:

```
ECHO OFF
CLS
IF "%1"=="AUTOEXEC.BAT" ECHO %%1 EQUALS THE AUTOEXEC.BAT
```

Now, once again, take an opposite of IF and make it IF NOT. **IF NOT** StringOne==StringTwo. Here's an example using the example above:

```
ECHO Off
CLS
IF NOT %1==AUTOEXEC.BAT ECHO %%1 IS NOT THE AUTOEXEC.BAT
```

Now, let's get back to what maybe the most used function the **IF ERRORLEVEL**. This command is tough to explain. Every time a command is performed, an errorlevel is returned. An errorlevel is a number returned when something happens. Let's say you wanted to make a DISKCOPY A: B:. The value returned for a successful disk copy is 0, a copy unsuccessful due to nonfatal disk error; 1, Copy incomplete because user pressed Ctrl-C; 2, Copy unsuccessful due to fatal disk error; 3, Insufficient memory or invalid drive; 4. Let's use IF ERRORLEVEL in an example:

```
ECHO OFF
CLS
DISKCOPY A: B:
IF ERRORLEVEL 4 ECHO INSUFFICIENT MEMORY OR THERE IS AN INVALID DRIVE
IF ERRORLEVEL 3 ECHO THERE'S A FATAL DISK ERROR THAT CAUSED AN UNSUCCESSFUL COPY
IF ERRORLEVEL 2 ECHO IF YOU WANT TO DO A DISKCOPY DON'T PRESS CONTROL-C
IF ERRORLEVEL 1 ECHO A NONFATAL DISK ERROR CAUSED AN UNSUCCESSFUL DISKCOPY
IF ERRORLEVEL 0 ECHO DISKCOPY  WAS SUCCESSFUL!
```

Here are some errorlevels that are useful in all causes. This file supposes you have MS-DOS 6. Here are the errorlevels:

Command	**Errorlevel**	**Meaning**
CHKDSK	0	No errors
	255	Errors detected
CHOICE	0	User pressed Ctrl-C
	255	Errors detected
DEFRAG	0	Successful defragmentation
	1	Internal error in processing
	2	No free clusters; DEFRAG needs at least one
	3	User pressed Ctrl-C; DEFRAG incomplete
	4	General error in processing

	5	Disk read error
	6	Disk write error
	7	Cluster allocation error; use CHKDSK /F
	8	Memory allocation error
	9	Insufficient memory
DELTREE	0	Successful deletion of directory and contents
DISKCOMP	0	Disks compare exactly
	1	Disks are not the same
	2	User press Ctrl-C; DISKCOMP incomplete
	3	Unrecoverable read or write error
	4	Insufficient memory, invalid drive, or syntax error
DISKCOPY	0	Successful copy
	1	Nonfatal read or write error
	2	User pressed Ctrl-C; DISKCOPY incomplete
	3	Unable to either read source or format target disk
	4	Insufficient memory, invalid drive, or syntax error
FIND	0	Successful search (match found)
	1	Search Completed (no matches found)
	2	Error during search
FORMAT	0	Successful format
	3	User pressed Ctrl-C; format incomplete
	4	Fatal error; format incomplete
	5	User pressed N at the continue prompt
MOVE	0	Files moved(or directory renamed) successfully
	1	Error in moving files or renaming directory
MSAV	86	Virus detected
REPLACE	0	Successful replacement
	2	Source file not found
	3	Source or target path not found
	5	Read-Only target file
	8	Insufficient memory
	11	Invalid command line
RESTORE	0	Successful restore
	1	No files found
	3	User pressed Ctrl-C; restore incomplete
	4	Fatal error; restore incomplete
SETVER	0	Successful update of version table
	1	Invalid command-line switch
	2	Invalid filename
	3	Insufficient memory
	4	Invalid version number
	5	Entry specified not found in table
	6	MS-DOS System files not found
	7	Invalid drive
	8	Too many command line parameters
	9	Missing command line parameter

	10	Error reading MS-DOS System files
	11	Version table corrupted in MS-DOS System files
	12	MS-DOS system files don't support version table
	13	Insufficient space in version table
	14	Error writing to MS-DOS System files
XCOPY	0	Successful copy
	1	No files found to copy
	2	User pressed Ctrl-C; copy incomplete
	4	Initialization error (not enough memory, invalid drive, file or path not found, or syntax error)
	5	Disk write error

Calling and invoking another batch file is easy with the call command. Use the syntax **Call** *BatchFile [Parameters]*. This is very useful to make fast, timesaving batch files that are interdependent.

Our next command will be **GOTO**. Goto is used usually with IF. Use GOTO in the following *If [Condition]* **GOTO** *Label*. The IF CONDITION is like IF ERRORLEVEL , IF EXIST, etc. The label in a section in the batch file that is preceded with a colon (:). If you're confused, look at the following example:

```
ECHO OFF
CLS
DISKCOPY A: B:
IF ERRORLEVEL 0 GOTO GOOD
GOTO END
:GOOD
ECHO SUCCESSFUL DISKCOPY!
:END
```

The final command we are going to learn is the **CHOICE**. The choice command uses errorlevels and is used with the following syntax **CHOICE /C**:*Keys [Message]*. Let's look at an example. Remember, each key is assigned an errorlevel. The first key is one, etc.:

```
ECHO OFF
:LOOP
CLS
ECHO                MAIN MENU
ECHO    A           MSAV
ECHO    B           DEFRAG
ECHO    C           MEM
ECHO    D           QUIT TO DOS
CHOICE /C:ABCD ENTER CHOICE:
IF ERRORLEVEL 4 GOTO QUIT
IF ERRORLEVEL 3 GOTO MEM
IF ERRORLEVEL 2 GOTO DEFRAG
IF ERRORLEVEL 1 GOTO MSAV
GOTO DONE
MEM:
MEMMAKER
GOTO DONE
:DEFRAG
DEFRAG
:MSAV
MSAVE
:QUIT
:DONE
```

The next operator is **For**, used for repeating tasks. This is similar in many programming languages to produce repetition. The syntax is:

For %%*batchvariable* **in (***list***) DO** *DOSCommand*

The batch variable can be any single letter. The variable is nothing, it is used for manipulation which will be shown in the following example. The list is a list of files. You can use wild cards such as * or ?. The DOS command is any command that you wish. The following example may produce some insight:

```
echo off
cls
REM BATTYPE.BAT -- A batch file for type batch files
FOR %%V IN (*.BAT) DO TYPE %%V
```

The "V" is a dummy parameter, a parameter with no meaning. The "(*.BAT)" is every file with the BAT extension. The "DO TYPE %%V" means type every file with the BAT extension.

The File Menu

New:
Closes all your open files and lets you make a new file.

Open:
Gives you a dialog box that lets you open a file

Save As:
Gives you the opportunity to save your file as something.

Print:
This menu option allows you to print your batch file.

Exit:
Exits the program

Add ANSI to your programs

Once you learn the commands of batch files, you want to move on to ANSI. ANSI stands for American National Standards Institute. ANSI actually adds color and capabilities that make your porgram much more interesting. To use ANSI, you must have the following in your CONFIG.SYS:
DEVICE=*pathname*\ANSI.SYS
or
DEVICEHIGH=*pathname*\ANSI.SYS

To use ANSI in BatchEdit, you must use something called the escape character. The escape character is a special that looks like this "<-". The character is used with the echo command. To insert the character in BatchEdit's environment, press the CTRL key. The following few paragraphs will provide a reference for BatchEdit's use of ANSI.

ANSI is useful for adding color. Use ANSI to insert color with the following context:
echo <- [*color***m***message*
Color is representative of a number. Here is the following numbers that are colors:

Color	Attribute
1	Bold Text
2*	Low intensity text
3*	Italic Text
4	Underscore on for IBM monochrome; underscore color (blue) for VGA
5	Blinking text
6*	Rapid-Blinking text
7	Reverse video text
8	Concealed text
31	Black foreground
32	Green foreground
33	Yellow foreground
34	Blue foreground
35	Magenta foreground
36	Cyan foreground
37	White foreground
40	Black foreground
41	Red foreground
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White foreground
48*	Subscript
49*	Superscript

* Not operative on VGA monitors

Adding Sparkle to Your Batch Files

The custom menu gives you the ability to change the color and font of your program. Just follow the directions and go through the motions and viola!, you have your sparkled batch file.

Running Your Batch Files

In the run menu, click on Go! Or press F5. You will be asked for any command-line parameters. Enter them, if any, as if you were typing them on the command-line. The second menu item you will see is "Text Editor". This takes you to MS-DOS Text Editor where you can open large files that you can't open in BatchEdit.

The Edit Commands:

Edit Cut:
    Cut out and copy the selected text to clipboard
Edit Copy:
    Keeps the selected text where it is but copy to the clipboard
Edit Delete:
    Deletes the selected text
Edit Paste:
    Pastes text to the current cursor position

The Debug Menu

Debugging your batch file is easier than you think. New debug commands are Find. Find asks you for a word and you enter it. BatchEdit will highlight it in your text.

The Debug List is a list of FindWords, or words that you are trying to find. You can and change the list. Use the add and cancel buttons to manipulate the Debug List.

The "Special" Menu

The "Special" menu may be something you use a lot. First off in the special menu is the password feature. This adds a command line password to your program so when another user runs it, they have to type the batch file name and then the password. BatchEdit does all the work for you. All you have to do is type up the password and click on OK.

Also in the "special" menu is Write. Since BatchEdit does not have the ability to open larger files, Write can be used. Write allows you to view the some files that BatchEdit cannot. All you have to do is type in the full path name of the file you want to work with and click on OK and there you are.

The third menu option is Make COM File. This makes BatchEdit's BAT file a faster, easier to use COM file. The name of the COM file, by default is BAT1.COM. You can and should rename this before making another COM file.