# The World of DOS - Creating Batch Files



---

## Batch Files

What are batch files? Batch files are not programs, pre se, they are lists of command line instructions that are *batched* together in one file. For the most part, you could manually type in the lines of a batch file and get the same results, but batch files make this work easy. Batch files do not contain "compiled" code like C++ so they can be opened, copied and edited. They are usually used for simple routines and low-level machine instruction, but they can be very powerful. If you look in your C:\, C:\WINDOWS, or C:\WINNT folder you will see a multitude of .BAT, .SYS, .CFG, .INF and other types. These are all kinds of batch files. This may shock you, but while most applications are writen in Basic or C++ they sit on a mountain of batch files. Batch files are the backbone of the Windows operating system, delete them and you've effectively disabled the OS. There is a reason for this. The system batch files on each computer are unique the that computer and change each time a program is loaded. The operating system must have access to these files and be able to add and delete instructions from them.

### Creating Batch files

Simple instructions

1. Open a text editor like notepad(NOT word or wordpad)
2. Type or copy this text:
```
@ECHO OFF
ECHO.
ECHO This is a batch file
ECHO.
PAUSE
CLS
EXIT
```
3. Save this as **batchfile.bat**, make sure there is no .txt extension after the .bat
4. Double-click the file icon

This is a little batch file I wrote that I use every day. It deletes the cookies that get dumped to my hard drive every time I go online. I could set my browser preferences not to accept cookies, but sometimes cookies are useful. Some CGI pages are unusable with cookies, sometimes when you enter a password for a Website, the site uses a cookie to remember your password. I just do not need hundreds of cookie files taking up space after I close my browser. With this batch file, all I have to do is double-click it and it deletes my cookies. Feel free to cut and paste this code to your Notepad or Wordpad. Save it as `cookiekill.bat` on your Desktop.

```
cls
REM ****************************************
REM **Cookie Kill Program Will not work in NT**
REM ****************************************

deltree /y c:\windows\cookies\*.*
deltree /y c:\windows\tempor~1\*.*
pause
cls
REM Cookies deleted!
```

What does the batch file do? The first line has the command `cls`. `cls` clears the screen window of any previous data. The next three lines start with `REM` for "remark." Lines begining with `REM` do not contain commands, but instructions or messages that will be displayed for the user. The next two lines begin with the command `deltree`, `deltree` not only deletes files but directories and sub-directories. In this case the file is deleting the directory `cookies` and all the files inside. This directory is automatically rebuilt. The `deltree` has been passed the parameter `/y`, this informs the process to answer "YES" to any confirmation questions. Sometimes you type the `DEL` or one of its cousins, the system will ask "Are sure you want to do this?" setting `/y` answers these prompts without interupting the process. The `pause` command halts the process temporarily and shows the users a list of all the files being deleted. `cls` clears the screen again, another `REM` line tells the user that the files are deleted. The last line contains only `:end` and returns the process to the command prompt. This version was created to show the user everything that is taking place in the process. The version bellow does the same thing without showing the user any details.

```
cls
@echo off

deltree /y c:\windows\cookies\*.*
deltree /y c:\windows\tempor~1\*.*

cls
```

Without REM lines there are no comments. The @echo off command keeps the process from being "echoed" in the DOS window, and without the pause and :end lines, the process runs and exits without prompting the user. In a process this small it is okay to have it be invisible to the user. With more a complex process, more visual feedback is needed. In computing there is fine line between too much and too little information. When in doubt give the user the oportunity to see what is going on.

**This version is a little more thurough, deletes alot of junk**

```
cls
@ECHO OFF
ECHO. ***********************************
ECHO. ** Clean Cookies and Temp Files **
ECHO. ** Will not work in NT **
ECHO. ******************************
deltree /y c:\windows\cookies\*.*
deltree /y c:\windows\tempor~1\*.*
deltree /y c:\progra~1\Netscape\Users\default\Cache\*.jpg
deltree /y c:\progra~1\Netscape\Users\default\Cache\*.gif
deltree /y c:\progra~1\Netscape\Users\default\Cache\*.htm
deltree /y c:\progra~1\Netscape\Users\default\archive\*.htm
deltree /y c:\progra~1\Netscape\Users\default\archive\*.gif
deltree /y c:\progra~1\Netscape\Users\default\archive\*.jpg
deltree /y c:\windows\temp\*.*
```

```
deltree /y c:\temp\*.*
deltree /y c:\windows\Recent\*.*
deltree /y c:\recycled\*.*
cls
EXIT
```

"C:\windows\history\today" will rebuld itself if you delete it. It's not a file, it's a specially configured directory structure that DOS doesn't see the same way that windows does. C:\windows\history\today doesn't actually exist as DOS sees it. Go into the C:\windows\history directory and type DIR/A this will show you the hidden directories and how they are named.

**WINNT Version**

```
@ECHO OFF
ECHO ************************************************
ECHO ** DEL replaces DELTREE, /Q replaces /Y **
ECHO ************************************************

del /Q c:\docume~1\alluse~1\Cookies\*.*
REM Change alluse~1 in the above line to your userID
del /q c:\winnt\temp\*.*
del /q c:\temp\*.*
del /q c:\winnt\Recent\*.*
del /q c:\*.chk
EXIT
```

**Add these lines for XP** - Provided by Patrick R.

```
del /q C:\Windows\Temp\Adware\*.*
del /q C:\Windows\Temp\History\*.*
del /q C:\Windows\Temp\Tempor~1\*.*
del /q C:\Windows\Temp\Cookies\*.*
```

One thing I do quite often is erase old floppy disks. I might have a stack of them and I don't care what's on them, but I want all the files gone including potential virii(everyone says "viruses" but "virii" is the proper term. Snob!). But I get tired of opening a DOS prompt and typing in the command to format the disk. So I wrote a one line batch file that does it for me. Save it as: "disk_wipe.bat"

```
format a: /u
```

Put a disk in the drive and double-click the .bat file icon.

**Batch File Utilities and Commands**

Any valid DOS command may be placed in a batch file, these commands are for setting-up the structure and flow of a batch file.

**CLS**
Clears the screen

---

**EXIT**
Exits the command-line process when the batch file terminates

```
EXIT
```

---

**BREAK**
When turned on, batch file will stop if the user presses < Ctrl >-< Break > when turned off, the script will continue until done.

```
BREAK=ON
```

```
BREAK=OFF
```

---

**CALL**
Calls another batch file and then returns control to the first when done.

```
CALL C:\WINDOWS\NEW_BATCHFILE.BAT
```

**Call another program**

```
CALL C:\calc.exe
```

Details.

---

**CHOICE**
Allows user input. Default is Y or N.
You may make your own choice with the /C: switch. This batch file displays a menu of three options. Entering 1, 2 or 3 will display a different row of symbols. Take note that the IF ERRORLEVEL statements must be listed in the reverse order of the selection. CHOICE is not recognized in some versions of NT.

```
@ECHO OFF
ECHO 1 - Stars
ECHO 2 - Dollar Signs
ECHO 3 - Crosses


CHOICE /C:123

IF errorlevel 3 goto CRS
IF errorlevel 2 goto DLR
IF errorlevel 1 goto STR

:STR
ECHO ******************
ECHO.
PAUSE
CLS
EXIT

:DLR
ECHO $$$$$$$$$$$$$$$$$$
ECHO.
PAUSE
CLS
EXIT

:CRS
ECHO ++++++++++++++++++++
ECHO.
PAUSE
CLS
EXIT
```

---

**FOR...IN...DO**
Runs a specified command for each file in a set of files. FOR *%%dosvar* IN (*set of items*) DO *command or command strcuture*.
**%%dosvar** is the variable that will hold items in the list, usually a single leter: %%a or %%b. Case sensitive, %%a is different from %A. The items in the **(set)** are assigned to this variable each time the loop runs.

**(set of items)** is one item or multiple items seperated by commas that determine how many times the loop runs.

**command or command strcuture** is the operation you want to perform for each item in the list.

This code will run through the set **(A, B, C)**, when it gets to B it will print the message: "B is in the set!"

```
```

```
FOR %%b in (A, B, C) DO IF %%b == B echo B is in the set!
```

This line will print the contents of C:\windows\desktop

```
FOR %%c in (C:\windows\desktop\*.*) DO echo %%c
```

So, you may create your own list or use various objects like files to determine the loop run. Details.

---

**GOTO**

To go to a different section in a batch file. You may create different sections by preceding the name with a colon.

```
:SUBSECTION
```

Programmers may find this similar to funtions or sub-routines.

```
@ECHO OFF
:FIRSTSECTION
ECHO This is the first section
PAUSE
GOTO SUBSECTION

:SUBSECTION
ECHO This is the subsection
PAUSE
```

**Skip sections of a batch file**

```
@ECHO OFF
:ONE
ECHO This is ONE, we'll skip TWO
PAUSE
GOTO THREE

:TWO
ECHO This is not printed

:THREE
ECHO We skipped TWO!
PAUSE
GOTO END
:END
CLS
EXIT
```

**Looping with GOTO**

```
:BEGIN
REM Endless loop, Help!!
GOTO BEGIN
```

Use with CHOICE

---

**IF, IF EXIST, IF NOT EXIST**

```
IF EXIST C:\tempfile.txt
DEL C:\tempfile.txt
```

```
IF NOT EXIST C:\tempfile.txt
COPY C:\WINDOWS\tempfile.txt C:\tempfile.txt
```

**Use with "errorlevel"**

The generic paramater **errorlevel** refers to the output another program or command and is also used with the CHOICE structure. If you try and run a command in a batch file and produces an error, you can use **errorlevel** to accept the returned code and take some action. For example, let's say you have a batch file that deletes some file.

```
COPY C:\file.txt C:\file2.txt
```

If "file.txt" doesn't exist, you will get the error: COULD NOT FIND C:\FILE.TXT. Instead, use a structure like this to create the file, then copy it by accepting the error.

```
@ECHO OFF
:START
COPY file.txt file2.txt
IF errorlevel 1 GOTO MKFILE
GOTO :END

:MKFILE
ECHO file text>file.txt
GOTO START

:END
ECHO Quitting
PAUSE
```

an errorlevel of 1 means there was an error, errorlevel of 0 means there was no error. You can see these levels by adding this line after any line of commands:

```
ECHO errorlevel: %errorlevel%
```

Details.

---

**PAUSE**
Pauses until the user hits a key.

This displays the familiar "Press any key to continue..." message.

---

**REM**
Allows a remark to be inserted in the batch script.

```
REM DIR C:\WINDOWS Not run as a command
DIR C:\WINDOWS     Run as a command
```

---

**ECHO**
Setting ECHO "on" will display the batch process to the screen, setting it to "off" will hide the batch process.

```
@ECHO OFF Commands are NOT displayed
@ECHO ON  Commands are displayed
```

ECHO can also be used in batch file to send output to the screen:

```
@ECHO OFF
ECHO.
ECHO Hi, this is a batch file
ECHO.
PAUSE
```

ECHO. sends a blank line.

To echo special characters, precede them with a caret:

ECHO ^<
ECHO ^>
Otherwise you will get an error.

The **@** before ECHO OFF suppresses the display of the initial ECHO OFF command. Without the @ at the beginning of a batch file the results of the ECHO OFF command will be displayed. The @ can be placed before any DOS command to suppress the display.

**Breaking long lines of code**
You may break up long lines of code with the caret ^. Put it at the end of a line, the next line must have space at the begining. Example:

```
copy file.txt file2.txt
```

would be:

```
```

```
copy file.txt^
  file2.txt
```

---

**SET**
Use to view or modify environment variables. More.

---

**LASTDRIVE**
Sets the last drive in the system.

```
lastdrive=Q
```

---

**MSCDEX**
Loads the CD-ROM software extensions(drivers), usually so an operating system can be then loaded from CD. See the AUTOEXEC.BAT section for special instructions concerning CD ROM installation.
Installing windows from a CD when the CDROM is not yet configured

---

**The AUTOEXEC.BAT file**

AUTOEXEC.BAT stands for *automatic execution batch file*, as in start-up automatically when the computer is turned on. Once a very important part of the operating system, it is being less used and is slowly disapearing from Windows. It is still powerful and useful. In NT versions it is called AUTOEXEC.NT, click here for more information.

Before the graphical user interface(GUI, "gooey") of Windows, turning on a PC would display an enegmatic C:\> and not much else. Most computer users used the same programs over-and-over, or only one program at all. DOS had a batch file which set certain system environments on boot-up. Because this was a batch file, it was possible to edit it and add a line to start-up the user's programs automatically.

When the first version of Windows was released users would turn their PCs on, and then type: WIN or WINDOWS at the prompt invoking the Windows interface. The next version of Windows added a line to the AUTOEXEC to start Windows right away. Exiting from Windows, brought one to the DOS prompt. This automatic invocation of Windows made a lot of people mad. Anyone who knew how to edit batch files would remove that line from the AUTOEXEC to keep Windows from controling the Computer. Most users do not even know that DOS is there now and have never seen it as Windows hides the any scrolling DOS script with their fluffy-cloud screen. At work I will often have to troubleshoot a PC by openning a DOS shell, the user's often panic, believing that I have broken their machine because the screen "turns black".

Most current versions of Windows have a folder called "Start-up." Any program or shortcut to a program placed in this folder will start automatically when the computer is turned on. This is much easier for most users to handle than editing batch files.

Old versions of DOS had a AUTOEXEC that looked like this:

```
@echo off
prompt $p$g
```

All this really did way set the DOS prompt to ">"

Later versions looked like this:

```
cls
@echo off
path c:\dos;c:\windows
set temp=c:\temp
Lh mouse
Lh doskey
Lh mode LPT1 retry
```

This AUTOEXEC.BAT loads DOS & then Windows. Sets up a "temp" directory. Loads the mouse driver, sets DOSKEY as the default and sets the printer retry mode. "Lh" stands for *Load High*, as in high memory.

**An AUTOEXEC.BAT from a Windows 3.11 Machine**

```
@ECHO On
rem C:\WINDOWS\SMARTDRV.EXE
C:\WINDOWS\SMARTDRV.EXE 2038 512
PROMPT $p$g
PATH C:\DOS;C:\WINDOWS;C:\LWORKS;C:\EXPLORER.4LC
SET TEMP=C:\DOS
MODE LPT1:,,P >nul
C:\DOS\SHARE.EXE /F:150 /L:1500
C:\WINDOWS\mouse.COM /Y
cd windows
WIN
```

This version simply sets DOS to boot to Windows.

```
SET HOMEDRIVE=C:
SET HOMEPATH=\WINDOWS
```

Whenever a program is installed on a computer, the setup program or wizard will often edit the AUTOEXEC. Many developer studios will have to "set a path" so programs can be compiled or run from any folder. This AUTOEXEC is an example of that:

```
SET PATH=C:\FSC\PCOBOL32;C:\SPRY\BIN
SET PATH=C:\Cafe\BIN;C:\Cafe\JAVA\BIN;%PATH%
SET HOMEDRIVE=C:
SET HOMEPATH=\WINDOWS
```

This AUTOEXEC sets the path for COBOL and JAVA development BINs. This way, the computer knows where to look for associated files for COBOL and JAVA files if they are not located directly in a BIN folder.

**Sets all the devices and boots to Windows.**
When the "REM" tags are removed the device commands become visible.

```
@SET PATH=C:C:\PROGRA~1\MICROS~1\OFFICE;%PATH%
REM [Header]
@ECHO ON
REM [CD-ROM Drive]
REM MSCDEX.EXE /D:OEMCD001 /L:Z
REM [Display]
REM MODE CON: COLS=80 LINES=25
REM [Sound, MIDI, or Video Capture Card]
REM SOUNDTST.COM
REM [Mouse]
REM MOUSE.COM
REM [Miscellaneous]
REM FACTORY.COM
```

**For loading Windows from a CD**

```
@echo off
MSCDEX.EXE /D:OEMCD001 /L:D
d:
cd \win95
oemsetup /k "a:\drvcopy.inf"
```

**For loading CDROM drivers**
Removing the "REM" tags uncomments the commands and runs them.

```
REM MSCDEX.EXE /D:OEMCD001 /l:d
```

```
REM MOUSE.EXE
```

**AUTOEXEC in NT**

NT does not use AUTOEXEC.BAT, the file is called **AUTOEXEC.NT** and should be found in the **C:\WINNT\system32** folder. Here is a sample AUTOEXEC.NT file:

```
@echo off

REM AUTOEXEC.BAT is not used to initialize the MS-DOS environment.
REM AUTOEXEC.NT is used to initialize the MS-DOS environment unless a
REM different startup file is specified in an application's PIF.

REM Install CD ROM extensions
lh %SystemRoot%\system32\mscdexnt.exe

REM Install network redirector (load before dosx.exe)
lh %SystemRoot%\system32\redir

REM Install DPMI support
lh %SystemRoot%\system32\dosx
SET PCSA=C:\PW32
dnp16.exe
```

***.NT and *.CMD**

.NT and .CMD may be used as .BAT files were used in earlier versions of Windows. You may notice on NT systems that there are fewer and fewer .BAT files. Try seaching for .NT or .CMD and you will find many of the same types of batch files that were available as .BATs. For example: **CONFIG.NT** has a similar function to the old **CONFIG.SYS** of Windows.

---

**CONFIG.SYS**

In Windows systems config.sys is used to set the initial values of the environment variables. To see your current settings, type SET on a command line. In early versions config.sys is a text file you can edit. In later versions it is a complied file that cannot be changed in a text editor. In newer NT versions it is not used at all. Try msconfig.exe instead.

```
REM [Header]
FILES=20
BUFFERS=20
DOS=HIGH,UMB
REM [SCSI Controllers]
REM DEVICE=SCSI.SYS
REM [CD-ROM Drive]
REM DEVICE=CDROM.SYS /D:OEMCD001
REM [Display]
REM DEVICE=DISPLAY.SYS
REM [Sound, MIDI, or Video Capture Card]
REM DEVICE=SOUND.SYS
REM [Mouse]
REM DEVICE=MOUSE.SYS
REM ------------------
REM [Miscellaneous]
REM DEVICE=SMARTDRV.EXE
```

---

**Types of "batch" files in windows**

**INI**, *.ini - Initalization file. These set the default variables for the system and programs. More

**CFG**, *.cfg - Configuration files.

**SYS**, *.sys - System files, can sometimes be edited, mostly compiled machine code in new versions. More.

**COM**, *.com - Command files. These are the executable files for all the DOS commands. In early

versions there was a seperate file for each command. Now, most are inside COMMAND.COM.

**NT**, *.nt - Batch files used by NT operating systems. More.

**CDM**, *.cmd - Batch files used in NT operating systems. More.

**Answer Files and Unattended Installations**
Customizing Unattended Installations
Answer Files
Customizing and Automating Installations
Automate Windows Installations

---

**Batch File Parameters**

You may put and use command-line parameters into your batch-files.

Suppose you had a batchfile called "test.bat" and these were the contents:

```
@echo off
if (%1) == (Hi) echo %1
```

and at the command line you entered: **test.bat Hi**, the output would be "Hi". If you entered **test.bat bye** you would get no response because the parameter did not match. the "**%1**" refers to the first parameter on the command line after the batch file name. If you want to two parameters, the script would look like this:

```
@echo off
if (%1) == (Hi) echo %1 %2
```

You could also just spit out what someone types in without a condition:

```
@echo off
echo %1 %2 %3 %4 %5 %6
```

Then typing **test.bat dont tell me what to do** would produce
dont tell me what to do because it is set up to handle 6 parameters and there are six words. You can tease someone by changing the order:

```
@echo off
echo %6 %3 %1 %2 %5 %4
```

**do me dont tell to what**

**Making your own variables**
You may use the **SET** command to create your own internal paramaters. This batch file:

```
@echo off
set myvar=Hi Joe
echo %myvar% is myvar
```

Will print **Hi Joe is myvar.** Notice a few important points. when we initialize **myvar** there are no **%** around it. When we use it, it must be between two **%**. Also, there must be no spaces between the **=** and the terms. When **myvar** is not in a **set** command or between **%** it is treated as a literal string.

You can make up your own parameter names and have many of them:

```
@echo off
set name=John Smith
set address=1 main street
set city=helltown

echo %name%
echo %address%
echo %city%
```

You could also assign command line parameters to the variables:

```
@echo off

set name=%1
set address=%2
set city=%3

echo %name%
echo %address%
echo %city%
```

The command line usually sees the space as a parameter delimiter, use double quotes " to make it ingore the spaces: **test.bat "Joe Smith" "1 Main Street" "Helltown"**.

Something important to remember about **SET**, it actaully creates a variable name in the file So if you enter **SET NAME=Joe** on the command line or in a batch file and then go to the command line and enter **ECHO %NAME%** the response will be **Joe**. Entering **SET** with no parameters will also show the whole list of **SET** variables. These will be erased when you reboot.

---

**The power of command line switches**

Most GUI programs have some kind of command line support which means you may automate their operation through batch files. For example, DOS does not have a built-in email sending function like UNIX. However, using an installed email program like Outlook, you may "force feed" the program on the command line. Outlook examples:**outlook /c ipm.note** will open a blank email, **outlook /c ipm.note /m msmith@yahoo.com** will open a blank email with the indicated address, **outlook /c ipm.note /a myfile.doc** attaches a file. More outlook switches, outlook programming.

An example using command line with winzip.

---

**The Windows Installation Catch-22**
You have a new computer with a unformated hard drive, or a drive with only DOS loaded. You want to load Windows from a CD, but you can't see the CD ROM from the DOS prompt. This is messy and can be screwed-up easily, luckily mistakes on this don't cause permanent damage. If you're lucky the CD ROM you have came with an installation disk(on floppy, of course). Putting this disk in and running the INSTALL.EXE or SETUP.EXE will install the drivers for you and alter the system files so you can load Windows from the CD ROM(Linux, by the way, has no problem with this!). If there is no INSTALL.EXE on the disk, you will have to edit lines in two files on your Windows 95 Boot/Install floppy disk. These files are: CONFIG.SYS and AUTOEXEC.BAT. Open these files for editing are look for lines that look like these:

```
REM DEVICE=CDROM.SYS /D:OEMCD001
```

And

```
REM C:\DOS\MSCDEX.EXE /D:OEMCD001
```

They may or may not be REMed out. You will need to change the "/D:OEMCD001" part of these lines to reflect the CD ROM that you have. For example if you have a Memorex it might be "/D:MSCD001". But be sure, check any manuals you might have lying around. If not, go to the manufacturer's website and down load the installation files. You will also need to figure out which drive letter it will be. If you only have on hard disk, it will be "D:" as in "/D:MSCD001," if you have two hard drives, or your drive is in several partitions, it might be "E:" or "F:". So then the line would be "/E:MSCD001" or "/F:MSCD001"

The Final line in CONFIG.SYS might be like this:

```
DEVICE=C:\WINDOWS\SBIDE.SYS /D:MSCD001 /V /P:170,15
```