

Batch file

A **batch file** is a script file in DOS, OS/2 and Microsoft Windows. It consists of a series of commands to be executed by the command-line interpreter, stored in a plain text file. A batch file may contain any command the interpreter accepts interactively and use constructs that enable conditional branching and looping within the batch file, such as IF, FOR, and GOTO labels. The term "batch" is from batch processing, meaning "non-interactive execution", though a batch file may not process a *batch* of multiple data.

Similar to Job Control Language (JCL), DCL and other systems on mainframe and minicomputer systems, batch files were added to ease the work required for certain regular tasks by allowing the user to set up a script to automate them. When a batch file is run, the shell program (usually COMMAND.COM or cmd.exe) reads the file and executes its commands, normally line-by-line.^[1] Unix-like operating systems, such as Linux, have a similar, but more flexible, type of file called a shell script.^[2]

The filename extension **.bat** is used in DOS and Windows. Windows NT and OS/2 also added **.cmd**. Batch files for other environments may have different extensions, e.g., **.btm** in 4DOS, 4OS2 and 4NT related shells.

The detailed handling of batch files has changed significantly between versions. Some of the detail in this article applies to all batch files, while other details apply only to certain versions.

Batch file



<u>Filename extensions</u>	<u>.bat</u> , <u>.cmd</u> , <u>.btm</u>
<u>Internet media type</u>	<u>application/bat</u> <u>application/x-bat</u> <u>application/x-msdos-program</u> <u>text/plain</u>
<u>Type of format</u>	<u>Scripting</u>
<u>Container for</u>	<u>Scripts</u>

Contents

Variants

DOS

Early Windows

OS/2

Windows NT

Filename extensions

Batch file parameters

Examples

Explanation

Limitations and exceptions

Null values in variables

Quotation marks and spaces in passed strings

Escaped characters in strings

Sleep or scripted delay

Text output with stripped CR/LF

[Setting a Uniform Naming Convention \(UNC\) working directory from a shortcut](#)

[Character set](#)

[**Batch viruses and malware**](#)

[**Other Windows scripting languages**](#)

[**See also**](#)


[**Notes**](#)

[**References**](#)

[**External links**](#)

Variants

DOS

In DOS, a batch file can be started from the [command-line interface](#) by typing its name, followed by any required parameters and pressing the  key. When DOS loads, the file [AUTOEXEC.BAT](#), when present, is automatically executed, so any commands that need to be run to set up the DOS environment may be placed in this file. Computer users would have the [AUTOEXEC.BAT](#) file set up the system date and time, initialize the DOS environment, load any resident programs or device drivers, or initialize network connections and assignments.

A .bat file name extension identifies a file containing commands that are executed by the command interpreter [COMMAND.COM](#) line by line, as if it were a list of commands entered manually, with some extra batch-file-specific commands for basic programming functionality, including a [GOTO](#) command for changing flow of line execution.

Early Windows

[Microsoft Windows](#) was introduced in 1985 as a [graphical user interface-based \(GUI\)](#) overlay on text-based [operating systems](#) and was designed to run on DOS. In order to start it, the [WIN](#) command was used, which could be added to the end of the [AUTOEXEC.BAT](#) file to allow automatic loading of Windows. In the earlier versions, one could run a .bat type file from Windows in the MS-DOS Prompt. [Windows 3.1x](#) and earlier, as well as [Windows 9x](#) invoked [COMMAND.COM](#) to run batch files.

OS/2

The [IBM OS/2](#) operating system supported DOS-style batch files. It also included a version of [REXX](#), a more advanced batch-file [scripting language](#). IBM and Microsoft started developing this system, but during the construction of it broke up after a dispute; as a result of this, IBM referred to their DOS-like console shell without mention of Microsoft, naming it just DOS, although this seemingly made no difference with regard to the way batch files worked from [COMMAND.COM](#).

OS/2's batch file interpreter also supports an [EXTPROC](#) command. This passes the batch file to the program named on the [EXTPROC](#) file as a data file. The named program can be a script file; this is similar to the [#!](#) mechanism.

Windows NT

Unlike Windows 98 and earlier, the Windows NT family of operating systems does not depend on MS-DOS. Windows NT introduced an enhanced 32-bit command interpreter (`cmd.exe`) that could execute scripts with either the `.CMD` or `.BAT` extension. `Cmd.exe` added additional commands, and implemented existing ones in a slightly different way, so that the same batch file (with different extension) might work differently with `cmd.exe` and `COMMAND.COM`. In most cases, operation is identical if the few unsupported commands are not used. `Cmd.exe`'s extensions to `COMMAND.COM` can be disabled for compatibility.

Microsoft released a version of `cmd.exe` for Windows 9x and ME called `WIN95CMD` to allow users of older versions of Windows to use certain `cmd.exe`-style batch files.

As of Windows 8, `cmd.exe` is the normal command interpreter for batch files; the older `COMMAND.COM` can be run as well in 32-bit versions of Windows able to run 16-bit programs.^[nb 1]

Filename extensions

.bat

The first filename extension used by Microsoft for batch files. This extension runs with DOS and all versions of Windows, under `COMMAND.COM` or `cmd.exe`, despite the different ways the two command interpreters execute batch files.

.cmd

Used for batch files in Windows NT family and sent to `cmd.exe` for interpretation. `COMMAND.COM` does not recognize this file name extension, so `cmd.exe` scripts are not executed in the wrong Windows environment by mistake. In addition, `append`, `dpath`, `ftype`, `set`, `path`, `assoc` and `prompt` commands, when executed from a `.bat` file, alter the value of the `errorlevel` variable only upon an error, whereas from within a `.cmd` file, they would affect `errorlevel` even when returning without an error.^[3] It is also used by IBM's OS/2 for batch files.

.btm

The extension used by 4DOS, 4OS2, 4NT and Take Command. These scripts are faster, especially with longer ones, as the script is loaded entirely ready for execution, rather than line-by-line.^[4]

Batch file parameters

`COMMAND.COM` and `cmd.exe` support that a number of special variables (`%0`, `%1` through `%9`) in order to refer to the path and name of the batch job and the first nine calling parameters from within the batch job, see also `SHIFT`. Non-existent parameters are replaced by a zero-length string. They can be used similar to environment variables, but are not stored in the environment. Microsoft and IBM refer to these variables as *replacement parameters* or *replaceable parameters*, whereas Digital Research, Novell and Caldera established the term *replacement variables*^[5] for them. JP Software calls them *batch file parameters*.^[6]

Examples

This example batch file displays `Hello World!`, prompts and waits for the user to press a key, and then terminates. (Note: It does not matter if commands are lowercase or uppercase unless working with variables)

```
@ECHO OFF
ECHO Hello World!
PAUSE
```

To execute the file, it must be saved with the filename extension suffix `.bat` (or `.cmd` for Windows NT-type operating systems) in plain text format, typically created by using a text editor such as Microsoft Notepad or a word processor working in plain text mode.

When executed, the following is displayed:

```
Hello World!  
Press any key to continue . . .
```

Explanation

The interpreter executes each line in turn, starting with the first. The `@` symbol at the start of any line prevents the prompt from displaying that command as it is executed. The command `ECHO OFF` turns off the prompt permanently, or until it is turned on again. The combined `@ECHO OFF` is often as here the first line of a batch file, preventing any commands from displaying, itself included. Then the next line is executed and the `ECHO Hello World!` command outputs `Hello World!`. The next line is executed and the `PAUSE` command displays `Press any key to continue . . .` and pauses the script's execution. After a key is pressed, the script terminates, as there are no more commands. In Windows, if the script is executed from an already running command prompt window, the window remains open at the prompt as in MS-DOS; otherwise, the window closes on termination.

Limitations and exceptions

Null values in variables

Variable expansions are substituted textually into the command, and thus variables which contain nothing simply disappear from the syntax, and variables which contain spaces turn into multiple tokens. This can lead to syntax errors or bugs.

For example, if `%foo%` is empty, this statement:

```
IF %foo%==bar ECHO Equal
```

parses as the erroneous construct:

```
IF ==bar ECHO Equal
```

Similarly, if `%foo%` contains `abc def`, then a different syntax error results:

```
IF abc def==bar ECHO Equal
```

The usual way to prevent this problem is to surround variable expansions in quotes so that an empty variable expands into the valid expression `IF ""=="bar"` instead of the invalid `IF ==bar`. The text that is being compared to the variable must also be enclosed in quotes, because the quotes are not special delimiting syntax; these characters represent themselves.

```
IF "%foo%"=="bar" ECHO Equal
```

The delayed !VARIABLE! expansion available in Windows 2000 and later may be used to avoid these syntactical errors. In this case, null or multi-word variables do not fail syntactically because the value is expanded after the IF command is parsed:

```
IF !foo!=="bar" ECHO Equal
```

Another difference in Windows 2000 or higher is that an empty variable (undefined) is not substituted. As described in previous examples, previous batch interpreter behaviour would have resulted in an empty string. Example:

```
C:\>set MyVar=
C:\>echo %MyVar%
%MyVar%

C:\>if "%MyVar%"==" " (echo MyVar is not defined) else (echo MyVar is %MyVar%)
MyVar is %MyVar%
```

Batch interpreters prior to Windows 2000 would have displayed result `MyVar is not defined`.

Quotation marks and spaces in passed strings

Unlike Unix/POSIX processes, which receive their command-line arguments already split up by the shell into an array of strings, a Windows process receives the entire command-line as a single string, via the `GetCommandLine` ([https://msdn.microsoft.com/en-us/library/windows/desktop/ms683156\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms683156(v=vs.85).aspx)) API function. As a result, each Windows application can implement its own parser to split the entire command line into arguments. Many applications and command-line tools have evolved their own syntax for doing that, and so there is no single convention for quoting or escaping metacharacters on Windows command lines.

- For some commands, spaces are treated as delimiters that separate arguments, unless those spaces are enclosed by quotation marks. Various conventions exist of how quotation marks can be passed on to the application:
 - A widely used convention is implemented by the command-line parser built into the Microsoft Visual C++ runtime library (<https://docs.microsoft.com/en-us/cpp/c-language/parsing-c-command-line-arguments>) in the `CommandLineToArgvW` (<https://docs.microsoft.com/en-us/windows/desktop/api/shellapi/nf-shellapi-commandlinetoargvw>) function. It uses the convention that $2n$ backslashes followed by a quotation mark (") produce n backslashes followed by a begin/end quote, whereas $(2n)+1$ backslashes followed by a quotation mark again produce n backslashes followed by a quotation mark literal. The same convention is part of the .NET Framework specification.^[7]
 - An undocumented aspect is that " " occurring in the middle of a quoted string produces a single quotation mark.^[7] (A CRT change in 2008 [msvcr90] modified this undocumented handling of quotes.^[8]) This is helpful for inserting a quotation mark in an argument without re-enabling interpretation of cmd metacharacters like |, & and >. (cmd does not recognize the usual \" as escaping the quote. It re-enables these special meanings on seeing the quote, thinking the quotation has ended.)
 - Another convention is that a single quotation mark (') is not included as part of the string. However, an escaped quotation mark ('"') can be part of the string.

- Yet another common convention comes from the use of Cygwin-derived ported programs. It does not differentiate between backslashes occurring before or not before quotes. See glob (programming) § Windows and DOS for information on these alternative command-line parsers.^[9]
- Some important Windows commands, like `cmd.exe` and `wscript.exe`, use their own rules.^[8]
- For other commands, spaces are not treated as delimiters and therefore do not need quotation marks. If quotes are included they become part of the string. This applies to some built-in commands like `echo`.

Where a string contains quotation marks, and is to be inserted into another line of text that must also be enclosed in quotation marks, particular attention to the quoting mechanism is required:

```
C:\>set foo="this string is enclosed in quotation marks"

C:\>echo "test 1 %foo%"
"test 1 "this string is enclosed in quotation marks""

C:\>eventcreate /T Warning /ID 1 /L System /SO "Source" /D "Example: %foo%"
ERROR: Invalid Argument/Option - 'string'.
Type "EVENTCREATE /?" for usage.
```

On Windows 2000 and later, the solution is to replace each occurrence of a quote character within a value by a series of three quote characters:

```
C:\>set foo="this string is enclosed in quotes"

C:\>set foo=%foo:="""%"

C:\>echo "test 1 %foo%"
"test 1 """"this string is enclosed in quotes""""

C:\>eventcreate /T Warning /ID 1 /L System /SO "Source" /D "Example: %foo%"
SUCCESS: A 'Warning' type event is created in the 'Source' log/source.
```

Escaped characters in strings

Some characters, such as pipe (|) characters, have special meaning to the command line. They cannot be printed as text using the `ECHO` command unless escaped using the caret ^ symbol:

```
C:\>Echo foo | bar
'bar' is not recognized as an internal or external command,
operable program or batch file.

C:\>Echo foo ^| bar
foo | bar
```

However, escaping does not work as expected when inserting the escaped character into an environment variable. The variable ends up containing a live pipe command when merely echoed. It is necessary to escape both the caret itself and the escaped character for the character display as text in the variable:

```
C:\>set foo=bar | baz
'baz' is not recognized as an internal or external command,
operable program or batch file.

C:\>set foo=bar ^| baz
C:\>echo %foo%
```

```
'baz' is not recognized as an internal or external command,  
operable program or batch file.
```

```
C:\>set foo=bar ^^^| baz  
C:\>echo %foo%  
bar | baz
```

The delayed `!VARIABLE!` expansion available with `CMD /V:ON` or with `SETLOCAL ENABLEDELAYEDEXPANSION` in Windows 2000 and later may be used to show special characters stored in environment variables because the variable value is expanded after the command was parsed:

```
C:\>cmd /V:ON  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
  
C:\>set foo=bar ^| baz  
C:\>echo !foo!  
bar | baz
```

Sleep or scripted delay

Until the `TIMEOUT` command was introduced with Windows Vista, there was no easy way to implement a timed pause, as the `PAUSE` command halts script activity indefinitely until any key is pressed.

Many workarounds were possible,^[10] but generally only worked in some environments: The `CHOICE` command was not available in older DOS versions, `PING` was only available if TCP/IP was installed, and so on. No solution was available from Microsoft, but a number of small utility programs, could be installed from other sources. A commercial example would be the 1988 Norton Utilities `Batch Enhancer` (BE) command, where `BE DELAY 18` would wait for 1 second, or the free 94-byte `WAIT.COM`^[11] where `WAIT 5` would wait for 5 seconds, then return control to the script. Most such programs are 16-bit .COM files, so are incompatible with 64-bit Windows.

Text output with stripped CR/LF

Normally, all printed text automatically has the control characters for carriage return (CR) and line feed (LF) appended to the end of each line.

■ batchtest.bat

```
@echo foo  
@echo bar
```

```
C:\>batchtest.bat  
foo  
bar
```

It does not matter if the two `echo` commands share the same command line; the CR/LF codes are inserted to break the output onto separate lines:

```
C:\>@echo Message 1&@echo Message 2  
Message 1  
Message 2
```

A trick discovered with Windows 2000 and later is to use the special prompt for input to output text without CR/LF trailing the text. In this example, the CR/LF does not follow Message 1, but does follow Line 2 and Line 3:

- batchtest2.bat

```
@echo off
set /p ="Message 1"<nul
echo Message 2
echo Message 3
```

```
C:\>batchtest2.bat
Message 1Message 2
Message 3
```

This can be used to output data to a text file without CR/LF appended to the end:

```
C:\>set /p ="Message 1"<nul >data.txt
C:\>set /p ="Message 2"<nul >>data.txt
C:\>set /p ="Message 3"<nul >>data.txt
C:\>type data.txt
Message 1Message 2Message 3
```

However, there is no way to inject this stripped CR/LF prompt output directly into an environment variable.

Setting a Uniform Naming Convention (UNC) working directory from a shortcut

It is not possible to have a command prompt that uses a UNC path as the current working directory; e.g. `\\server\share\directory\`

The command prompt requires the use of drive letters to assign a working directory, which makes running complex batch files stored on a server UNC share more difficult. While a batch file can be run from a UNC file path, the working directory default is `C:\Windows\System32\`.

In Windows 2000 and later, a workaround is to use the **PUSHD** and **POPD** command with command extensions.^[nb 2]

If not enabled by default, command extensions can be temporarily enabled using the `/E:ON` switch for the command interpreter.

So to run a batch file on a UNC share, assign a temporary drive letter to the UNC share, and use the UNC share as the working directory of the batch file, a Windows shortcut can be constructed that looks like this:

- Target: `%COMSPEC% /E:ON /C "PUSHD ""\\SERVER\SHARE\DIR1\DIR2\""" & BATCHFILE.BAT & POPD"`

The working directory attribute of this shortcut is ignored.

This also solves a problem related to User Account Control (UAC) on Windows Vista and newer. When an administrator is logged on and UAC is enabled, and they try to run a batch file as administrator from a network drive letter, using the right-click file context menu, the operation will unexpectedly fail. This is because the

elevated UAC privileged account context does not have network drive letter assignments, and it is not possible to assign drive letters for the elevated context via the Explorer shell or logon scripts. However, by creating a shortcut to the batch file using the above **PUSHD / POPD** construct, and using the shortcut to run the batch file as administrator, the temporary drive letter will be created and removed in the elevated account context, and the batch file will function correctly.

The following syntax does correctly expand to the path of the current batch script.

```
%~dp0
```

UNC default paths are turned off by default as they used to crash older programs.^[12]

The **Dword** registry value **DisableUNCCheck** at **HKEY_CURRENT_USER\Software\Microsoft\Command Processor**^[12] allows the default directory to be UNC. **CD** command will refuse to change but placing a UNC path in Default Directory in a shortcut to **Cmd** or by using the **Start** command. **start "" /d \\127.0.0.1\C\$ "cmd /k"** (C\$ share is for administrators).

Character set

Batch files use an OEM character set, as defined by the computer, e.g. [Code page 437](#). The non-ASCII parts of these are incompatible with the [Unicode](#) or [Windows character sets](#) otherwise used in Windows so care needs to be taken.^[13] Non-English file names work only if entered through a DOS character set compatible editor. File names with characters outside this set do not work in batch files.

To get output in Unicode into file pipes from an internal command such as **dir**, one can use the **cmd /U** command. For example, **cmd /U /C dir > files.txt** creates a file containing a directory listing with correct Windows characters, in the [UTF-16LE](#) encoding.

Batch viruses and malware

As with any other programming language, batch files can be used maliciously. Simple [trojans](#) and [fork bombs](#) are easily created, and batch files can do a form of [DNS poisoning](#) by modifying the [hosts file](#). Batch viruses are possible, and can also spread themselves via [USB flash drives](#) by using Windows' [Autorun](#) capability.^[14]

The following command in a batch file will delete all the data in the current directory (folder) - without first asking for confirmation:

```
del /Q *.* *
```

These three commands are a simple [fork bomb](#) that will continually replicate itself to deplete available system resources, slowing down or crashing the system:

```
:TOP
start "" %0
goto TOP
```

Other Windows scripting languages

The cmd.exe command processor that interprets .cmd files is supported in all 32- and 64-bit versions of Windows up to at least Windows 10. COMMAND.EXE, which interprets .BAT files, was supported in all 16- and 32-bit versions up to at least Windows 10.^[nb 3]

There are other, later and more powerful, scripting languages available for Windows. However, these require the scripting language interpreter to be installed before they can be used:

- KiXtart (.kix) — developed by a Microsoft employee in 1991, specifically to meet the need for commands useful in a network logon script while retaining the simple 'feel' of a .cmd file.
- Windows Script Host (.vbs , .js and .wsf) — released by Microsoft in 1998, and consisting of cscript.exe and wscript.exe, runs scripts written in VBScript or JScript. It can run them in windowed mode (with the wscript.exe host) or in console-based mode (with the cscript.exe host). They have been a part of Windows since Windows 98.
- PowerShell (.ps1) — released in 2006 by Microsoft and can operate with Windows XP (SP2/SP3) and later versions. PowerShell can operate both interactively (from a command-line interface) and also via saved scripts, and has a strong resemblance to Unix shells.^[15]
- Unix-style shell scripting languages can be used if a Unix compatibility tool, such as Cygwin, is installed.
- Cross-platform scripting tools including Perl, Python, Ruby, Rexx, Node.js and PHP are available for Windows.

Script files run if the filename without extension is entered. There are rules of precedence governing interpretation of, say, DoThis if DoThis.com, DoThis.exe, DoThis.bat, DoThis.cmd, etc. exist; by default DoThis.com has highest priority. This default order may be modified in newer operating systems by the user-settable PATHEXT environment variable.

See also

- List of DOS commands

Notes

1. To verify that COMMAND.COM remains available (in the \WINDOWS\SYSTEM32 directory), type COMMAND.COM at the 32-bit Windows 7 command prompt.
2. "If Command Extensions are enabled the PUSH D command accepts network paths in addition to the normal drive letter and path. If a network path is specified, PUSH D creates a temporary drive letter that points to that specified network resource and then change the current drive and directory, using the newly defined drive letter. Temporary drive letters are allocated from Z: on down, using the first unused drive letter found." --The help for PUSH D in Windows 7
3. Availability of CMD.EXE and COMMAND.COM can be confirmed by invoking them in any version of Windows (COMMAND.COM not in 64-bit versions; probably only available in Windows 8 32-bit versions if installed with option to support 16-bit programs).

References

1. "Using batch files: Scripting; Management Services" ([https://technet.microsoft.com/en-us/library/cc758944\(WS.10\).aspx](https://technet.microsoft.com/en-us/library/cc758944(WS.10).aspx)). Technet.microsoft.com. 2005-01-21. Retrieved 2012-11-30.
2. Henry-Stocker, Sandra. "Use your Unix scripting skills to write a batch file" (<https://www.itworld.com/article/2817193/operating-systems/unix-tip--use-your-unix-scripting-skills-to-write-a-batch-file.html>). itworld.com. IT World. Retrieved 2018-06-13.

3. "Difference between bat and cmd | WWoIT - Wayne's World of IT" (<http://waynes-world-it.blogspot.fr/2008/08/difference-between-bat-and-cmd.html>). *waynes-world-it.blogspot.fr*. 2012-11-15. Retrieved 2012-11-30.
4. "btm file extension :: all about the .btm file type" (<http://www.cryer.co.uk/filetypes/b/btm.htm>). *Cryer.co.uk*. Retrieved 2012-11-30.
5. *Caldera DR-DOS 7.02 User Guide* (<https://web.archive.org/web/20161105114931/http://www.dr-dos.net/documentation/usergeng/uglontoc.htm>), Caldera, Inc., 1998 [1993, 1997], archived from the original (<http://www.dr-dos.net/documentation/usergeng/uglontoc.htm>) on 2016-11-05, retrieved 2013-08-10
6. Brothers, Hardin; Rawson, Tom; Conn, Rex C.; Paul, Matthias R.; Dye, Charles E.; Georgiev, Luchezar I. (2002-02-27). *4DOS 8.00 online help*.
7. ".NET Core Runtime: System.Diagnostics.Process.Unix" (<https://github.com/dotnet/runtime/blob/90a2d87/src/libraries/System.Diagnostics.Process/src/System.Diagnostics.Process.Unix.cs#L858-L861>). *GitHub*. Retrieved 2020-02-11. "Two consecutive double quotes inside an inQuotes region should result in a literal double quote (the parser is left in the inQuotes region). This behavior is not part of the spec of code:ParseArgumentsIntoList, but is compatible with CRT and .NET Framework."
8. Deley, David. "How Command Line Parameters Are Parsed" (<http://daviddeley.com/autohotkey/parameters/parameters.htm#WINCRULES>).
9. "Child process documentation, section Windows Command Line, NodeJS PR #29576" (https://github.com/Artoria2e5/node/blob/fix!/child-process-args/doc/api/child_process.md#windows-command-line). *GitHub*. Retrieved 2020-02-11.
10. "How to do a delay" (<http://www.ericphelps.com/batch/samples/sleep.txt>), *ericphelps.com*
11. Utilities for DOS, linking to WAIT.ZIP (archive of WAIT.COM) and other programs (http://www.uw-e-sieber.de/util_e.html)
12. <https://support.microsoft.com/en-us/kb/156276>
13. Chen, Raymond. "Keep your eye on the code page" (<http://blogs.msdn.com/b/oldnewthing/archive/2005/03/08/389527.aspx>). *Microsoft*.
14. <http://www.explorehacking.com/2011/01/batch-files-art-of-creating-viruses.html>
15. "Windows PowerShell - Unix comes to Windows" (<http://geekswithblogs.net/sdorman/archive/2006/06/18/82258.aspx>). *Geekswithblogs.net*. Retrieved 2012-11-30.

External links

- [Microsoft Windows XP Batch file reference](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490869(v%3dtechnet.10)) ([https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490869\(v%3dtechnet.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490869(v%3dtechnet.10)))
- [How Windows batch files work](http://commandwindows.com/batch.htm) (<http://commandwindows.com/batch.htm>)
- [Windows 10 batch file commands](https://ss64.com/nt/) (<https://ss64.com/nt/>)
- [FreeDOS' FreeCOM : complete feature list](http://www.freedos.org/freecom/) (<http://www.freedos.org/freecom/>)
- [MS-DOS+Win../95/98/ME batch programming links](https://web.archive.org/web/20120419153526/http://www.netikka.net/tsneti/http/tsnetihttpprog.php#batch) (<https://web.archive.org/web/20120419153526/http://www.netikka.net/tsneti/http/tsnetihttpprog.php#batch>)
- [Windows Command Line Interface script programming links](https://web.archive.org/web/20120419153526/http://www.netikka.net/tsneti/http/tsnetihttpprog.php#cmdscript) (<https://web.archive.org/web/20120419153526/http://www.netikka.net/tsneti/http/tsnetihttpprog.php#cmdscript>)
- [scripting related information \(also command line\)](http://www.robvanderwoude.com/) (<http://www.robvanderwoude.com/>)
- [dbenham. "How does the Windows Command Interpreter \(CMD.EXE\) parse scripts?"](https://stackoverflow.com/a/4095133) (<https://stackoverflow.com/a/4095133>). *Stack Overflow*.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Batch_file&oldid=1031582791"

This page was last edited on 2 July 2021, at 12:15 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.