

# Windows Batch Scripting: Getting Started

---

- [Overview](#)
- Part 1 – Getting Started
- [Part 2 – Variables](#)
- [Part 3 – Return Codes](#)
- [Part 4 – stdin, stdout, stderr](#)
- [Part 5 – If/Then Conditionals](#)
- [Part 6 – Loops](#)
- [Part 7 – Functions](#)
- [Part 8 – Parsing Input](#)
- [Part 9 – Logging](#)
- [Part 10 – Advanced Tricks](#)

## Getting Started with Windows Batch Scripting

Windows batch scripting is incredibly accessible – it works on just about any modern Windows machine. You can create and modify batch scripts on just about any modern Windows machine. The tools come out of the box: the Windows command prompt and a text editor like Notepad.exe. It's definitely far from the best shell scripting language, but, it gets the job done. It's my "duct tape" for Windows.

## Launching the Command Prompt

Windows gurus launch the command prompt using the keyboard shortcut **Windows Logo Key+R** (i.e., "Run") > Type **cmd.exe** then **Enter**. This is way faster than navigating the Windows Start Menu to find the Command Prompt.

## Editing Batch Files

The universal text editor for batch files is Notepad (**Windows Logo Key**

- **R** > Type **notepad** then **Enter**). Since batch files are just ASCII text, you can probably use just about any text editor or word processor. Very few editors do anything special for Batch files like syntax highlighting or keyword support, so notepad is good enough fine and will likely be installed on just about every Windows system you encounter.

## Viewing Batch Files

I would stick with Notepad for viewing the contents of a batch file. In Windows Explorer (aka, "My Computer"), you should be able to view a batch file in Notepad by right clicking the file and selecting **Edit** from the context menu. If you need to view the contents within a command prompt window itself, you can use a DOS command like **TYPE myscript.cmd** or **MORE myscript.cmd** or **EDIT myscript.cmd**

## Batch File Names and File Extensions


Assuming you are using Windows XP or newer, I recommend saving your batch files with the file extension `.cmd`. Some seriously outdated Windows versions used `.bat`, though I recommend sticking with the more modern `.cmd` to [avoid some rare side effects with .bat files](#).

With the `.cmd` file extension, you can use just about filename you like. I recommend avoiding spaces in filenames, as spaces only create headaches in shell scripting. Pascal casing your filenames is an easy way to avoid spaces (e.g., `HelloWorld.cmd` instead of `Hello World.cmd`). You can also use punctuation characters like `.` or `-` or `_` (e.g. `Hello.World.cmd`, `Hello-World.cmd`, `Hello_World.cmd`).

Another thing with names to consider is avoiding names that use the same name of any built-in commands, system binaries, or popular programs. For example, I would avoid naming a script `ping.cmd` since there is a widely used system binary named `ping.exe`. Things might get very confusing if you try to run `ping` and inadvertently call `ping.cmd` when you really wanted `ping.cmd`. (Stay tuned for how this could happen.) I might call the script `RemoteHeartbeat.cmd` or something similar to add some context to the script's name and also avoid any naming collisions with any other executable files. Of course, there could be a very unique circumstance in which you want to modify the default behavior of `ping` in which this naming suggestion would not apply.

## Saving Batch Files in Windows

Notepad by default tries to save all files as plain jane text files. To get Notepad to save a file with a `.cmd` extension, you will need to change the "Save as type" to "All Files (.\*)". See the screenshot below for an example of saving a script named "HelloWorld.cmd" in Notepad.

 Screenshot of saving a batch file in Notepad

**SIDEBAR:** I've used a shortcut in this screenshot that you will learn more about later. I've saved the file to my "user profile folder" by naming the file `%USERPROFILE%\HelloWorld.cmd`. The `%USERPROFILE%` keyword is the Windows environmental variable for the full path to your user profile folder. On newer Windows systems, your user profile folder will typically be `C:\Users\<your username>`. This shortcut saves a little bit of time because a new command prompt will generally default the "working directory" to your user profile folder. This lets you run `HelloWorld.cmd` in a new command prompt without changing directories beforehand or needing to specify the path to the script.

## Running your Batch File

The easy way to run your batch file in Windows is to just double click the batch file in Windows Explorer (aka "My Computer"). Unfortunately, the command prompt will not give you much of a chance to see the output and any errors. The command prompt window for the script will disappear as soon as the script exits. (We will learn how to handle this problem in [Part 10 – Advanced Tricks](#) ).

When editing a new script, you will likely need to run the batch file in an existing command window. For newbies, I think the easiest foolproof way to run your script is to drag and drop the script into a command prompt window. The command prompt will enter the full path to your script on the command line, and will quote any paths containing spaces.

Some other tips to running batch files:

- You can recall previous commands using the up arrow and down arrow keys to navigate the command line history.
- I usually run the script as `%COMPSPEC% /C /D "C:\Users\User\SomeScriptPath.cmd" Arg1 Arg2 Arg3` This runs your script in a new command prompt child process. The `/C` option instructs the child process to quit when your script quits. The `/D` disables any auto-run scripts (this is optional, but, I use auto-run scripts). The reason I do this is to keep the command prompt window from automatically closing should my script, or a called script, call the `EXIT` command. The `EXIT` command automatically closes the command prompt window unless the `EXIT` is called from a child command prompt process. This is annoying because you lose any messages printed by your script.

## Comments

The official way to add a comment to a batch file is with the `REM` (Remark) keyword:

```
REM This is a comment!
```

The power user method is to use `::`, which is a hack to uses the the label operator `:` twice, which is almost always ignored.

Most power authors find the `::` to be less distracting than `REM`. Be warned though there are a few places where `::` will cause errors.

```
:: This is a comment too!! (usually!)
```

For example, a `FOR` loop will error out with `::` style comments. Simply fall back to using `REM` if you think you have a situation like this.

## Silencing Display of Commands in Batch Files

The first non-comment line of a batch file is usually a command to turn off printing (ECHO'ing) of each batch file line.

```
@ECHO OFF
```

The `@` is a special operator to suppress printing of the command line. Once we set ECHO'ing to off, we won't need the `@` operator again in our script commands.

You restore printing of commands in your script with:

```
ECHO ON
```

---

Upon exit of your script, the command prompt will automatically restore ECHO to it's previous state.

## Debugging Your Scripts

Batch files involve a lot of trial and error coding. Sadly, I don't know of any true debugger for Windows batch scripts. Worse yet, I don't know of a way to put the command processor into a verbose state to help troubleshoot the script (this is the common technique for Unix/Linux scripts.) Printing custom ad-hoc debugging messages is about your only option using the `ECHO` command. Advanced script writers can do some trickery to selectively print debugging messages, though, I prefer to remove the debugging/instrumentation code once my script is functioning as desired.