

# Windows Batch Scripting: Stdin, Stdout, Stderr

---

- [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](#)

DOS, like Unix/Linux, uses the three universal “files” for keyboard input, printing text on the screen, and the printing errors on the screen. The “Standard In” file, known as stdin, contains the input to the program/script. The “Standard Out” file, known as stdout, is used to write output for display on the screen. Finally, the “Standard Err” file, known as stderr, contains any error messages for display on the screen.

## File Numbers

Each of these three standard files, otherwise known as the standard streams, are referenced using the numbers 0, 1, and 2. Stdin is file 0, stdout is file 1, and stderr is file 2.

## Redirection

A very common task in batch files is sending the output of a program to a log file. The `>` operator sends, or redirects, stdout or stderr to another file. For example, you can write a listing of the current directory to a text file:

```
DIR > temp.txt
```

The `>` operator will overwrite the contents of temp.txt with stdout from the DIR command. The `>>` operator is a slight variant that appends the output to a target file, rather than overwriting the target file.

A common technique is to use `>` to create/overwrite a log file, then use `>>` subsequently to append to the log file.

```
SomeCommand.exe > temp.txt  
OtherCommand.exe >> temp.txt
```

By default, the `>` and `>>` operators redirect stdout. You can redirect stderr by using the file number `2` in front of the operator:

```
DIR SomeFile.txt 2>> error.txt
```

You can even combine the stdout and stderr streams using the file number and the `&` prefix:

```
DIR SomeFile.txt 2>&1
```

This is useful if you want to write both stdout and stderr to a single log file.

```
DIR SomeFile.txt > output.txt 2>&1
```

To use the contents of a file as the input to a program, instead of typing the input from the keyboard, use the `<` operator.

```
SORT < SomeFile.txt
```

## Suppressing Program Output

The pseudofile `NUL` is used to discard any output from a program. Here is an example of emulating the Unix command `sleep` by calling ping against the loopback address. We redirect stdout to the `NUL` device to avoid printing the output on the command prompt screen.

```
PING 127.0.0.1 > NUL
```

## Redirecting Program Output As Input to Another Program

Let's say you want to chain together the output of one program as input to another. This is known as "piping" output to another program, and not suprisingly we use the pipe character `|` to get the job done. We'll sort the output of the `DIR` command.


```
DIR /B | SORT
```

## A Cool Party Trick

You can quickly create a new text file, say maybe a batch script, from just the command line by redirecting the command prompt's own stdin, called `CON`, to a text file. When you are done typing,

hit **CTRL+Z**, which sends the end-of-file (EOF) character.

```
TYPE CON > output.txt
```

 Screenshot of sending keyboard input to a file

There are a number of other special files on DOS that you can redirect, however, most are a bit dated like LPT1 for parallel port printers or COM1 for serial devices like modems.