

## How-to: Redirect

*command*  
*command*

*command*

*command*  
*command*

*command*

*command*

*command*

*command*

*command*

Success and failure  
In most cases the f

For clarity the syntax on this page has spaces before and after the redirection operators, in practice you may want to omit those to avoid additional space characters being added to the output. `Echo Demo Text> Demofile.txt`

Numeric handles:

STDIN = 0 Keyboard input  
STDOUT = 1 Text output  
STDERR = 2 Error text output  
UNDEFINED = 3-9 (In PowerShell 3.0+ these are defined)

When redirection is performed without specifying a numeric handle, the the default < redirection input operator is zero (0) and the default > redirection output operator is one (1). This means that '>' alone will not redirect error messages.

<i>command</i> 2> filename	Redirect any error message into a file
<i>command</i> 2>> filename	Append any error message into a file
( <i>command</i> )2> filename	Redirect any CMD.exe error into a file
<i>command</i> > file 2>&1	Redirect errors and output to one file
<i>command</i> > fileA 2> fileB	Redirect output and errors to separate files
<i>command</i> 2>&1 >filename	This will fail!

Redirect to NUL (hide errors)

<i>command</i> 2> nul	Redirect error messages to NUL
<i>command</i> >nul 2>&1	Redirect error and output to NUL
<i>command</i> >filename 2> nul	Redirect output to file but suppress error
( <i>command</i> )>filename 2> nul	Redirect output to file but suppress CMD.exe errors

Any long filenames must be surrounded in "double quotes".  
A CMD error is an error raised by the command processor itself rather than the program/command.

Some commands, (e.g. COPY) do not place all their error mesages on the error stream, so to capture those you must redirect both STDOUT and STDERR with *command* > file 2>&1

Redirection with > or 2> will overwrite any existing file.

You can also redirect to a printer with > PRN or >LPT1 or to the console with >CON

To prevent the > and < characters from causing redirection, escape with a caret: ^> or ^<

## Redirection - issues with trailing numbers

Redirecting a string (or variable containing a string) will fail to work properly if there is a single numeral at the end, anything from 0 to 9.  
e.g. this will fail:

```
Set _demo=abc 5
Echo %_demo%>>demofile.txt
```

One workaround for this is to add a space before the '>>' but that space will end up in the output.  
Moving the redirection to the end of the command works.

```
Set _demo=
>>demofile.txt
```

## Create a new file

Create an empty file

```
Type NUL > newfile.txt
or
Copy NUL E newfile.txt
or
BREAK > Emptyfile.txt
```

## Multiple commands

In a batch file, you can execute multiple commands on a single line, separated by a semicolon (;). This is useful for running a series of related commands together.

```
SET /P _c=Enter command:
This behavior will work in Windows 7 and later.
```

## Redirect multiple lines

```
Redirect multiple lines to a file
(
Echo sample text1
Echo sample text2
) > c:\logfile.txt
```

## Unicode

The CMD Shell can redirect ASCII/ANSI (the default) or Unicode (UCS-2 le) but not UTF-8. This can be selected by launching CMD /A or CMD /U

In Windows 7 and earlier versions of Windows, the redirection operator '>' would strip many Extended ASCII/Unicode characters from the output. Windows 10 no longer does this.

## Pipes and CMD.exe

You can redirect and execute a batch file into CMD.exe with:

```
CMD < sample.cmd
```

Surprisingly this will work with any file extension (.txt .xls etc) if the file contains text then CMD will attempt to execute it. No sanity checking is performed.

When a command is piped into any external command/utility ( *command* | *command* ) this will instantiate a new CMD.exe instance. e.g.

```
TYPE test.txt | FIND "Smith"
```

Is in effect running:

```
TYPE test.txt | cmd.exe /S /D /C FIND "Smith"
```

This has a couple of side effects:  
If the items being piped (the left hand side of the pipe) include any caret escape characters ^ they will need to be doubled up so that they survive into the new CMD shell.  
Any newline (CR/LF) characters in the first *command* will be turned into & operators. (see [StackOverflow](#))

On modern hardware, starting a new CMD shell has no noticable effect on performance.

For example, this syntax works, but would fail if the second or subsequent (piped) lines were indented with a space:

```
@Echo Off
Echo abc def |^
Find "abc" |^
Find "def"> outfile.txt
```

Multi-line single commands with lots of parameters, can be indented as in this example:

```
Echo abc def ^
    ghi jkl ^
    mno pqr
```

## Redirection anywhere

Although the redirection operator (>) can be placed anywhere in the command line, it is best practice to place it at the end of the command line. All of these commands will work:

```
Echo %date% >date.txt
Echo %date% >date.txt & Echo %date% >date2.txt
Echo>date.txt %date%
>C:\date.txt Echo %date%
```

All of them Echo the date to date.txt

If the command is indented, the redirection operator (>) captures the result of the SORT command.

Code like this will not work:

```
Set _out=output.txt
Echo %date% >_out
```

Will inadvertently create a file named output.txt

One solution is to use the trick below:

```
Echo %date% >>_out
```

This assumes the redirection operator (>) is not used elsewhere in the command line. If you do use the trick above to move the redirection operator to a location where it won't cause any trouble:

```
>schedule.txt Echo %_message%
```

Example via [Raymond Chen](#)

The idea of redirection anywhere in the line was first introduced in [version 2 of sh](#), written by Ken Thompson in 1972.

## Exit Codes

If the *filename* or *command* is not found then redirection will set an Exit Code of 1

When redirecting the output of DIR to a file, you may notice that the output file (if in the same folder) will be listed with a size of 0 bytes. The command interpreter first creates the empty destination file, then runs the DIR command and finally saves the redirected text into the file.

The maximum number of consecutive pipes is [2042](#)

## Examples

```
DIR >MyFileListing.txt

DIR /o:n >"Another list of Files.txt"

DIR C:\ >List_of_C.txt 2>errorlog.txt

DIR C:\ >List_of_C.txt & DIR D:\ >List_of_D.txt

ECHO y| DEL *.txt

ECHO Some text ^<html tag^> more text

COPY nul empty.txt

MEM /C >>MemLog.txt

Date /T >>MemLog.txt

SORT < MyTextFile.txt

SET _output=%_missing% 2>nul

FIND /i "Jones" < names.txt >logfile.txt

(TYPE logfile.txt >> newfile.txt) 2>nul
```

“Stupidity, outrage, vanity, cruelty, iniquity, bad faith, falsehood,  
we fail to see the whole array when it is facing in the same direction as we” ~ Jean Rostand (French Historian)

Related commands

[CON](#) - Console dev  
[conIN\\$](#) and [conO](#)  
[SORT](#) - Sort input.  
[CMD Syntax](#)  
[TYPE](#) - Display the  
[Command Redirec](#)  
[Successive redirec](#)  
[Equivalent PowerS](#)  
[Equivalent bash co](#)

( SS64 )

We value your privacy

We and our [Google](#) store and/or access information on a device, such as cookies and process personal data, such as unique identifiers and standard information sent by a device for personalised advertising and content, advertising and content measurement, audience research and services development. With your permission we and our partners may use precise geolocation data and identification through device scanning. You may click to consent to our and our 1414 partners' processing as described above. Alternatively you may click to refuse to consent or access more detailed information and change your preferences before consenting. Please note that some processing of your personal data may not require your consent, but you have a right to object to such processing. Your preferences will apply to this website only. You can change your preferences or withdraw your consent at any time by returning to this site and clicking the "Privacy" button at the bottom of the webpage. Please note that this website/app uses one or more Google services and may gather and store information including but not limited to your visit or usage behaviour. You may click to grant or deny consent to Google and its third-party tags to use your data for below specified purposes in below Google consent section.