How-to: Redirection

```
command > filename     Redirect command output to a file
command >&n            Redirect command output to the input of handle n

command >> filename    APPEND into a file

command < filename     Type a text file and pass the text to command.
command <&n            Read input from handle n and write it to command.

commandA | commandB    Pipe the output from commandA into commandB

commandA & commandB    Run commandA and then run commandB
commandA && commandB   Run commandA, if it succeeds then run commandB
commandA || commandB   Run commandA, if it fails then run commandB

commandA && commandB || commandC
```

If *commandA* succeeds run *commandB,* if *commandA* fails run *commandC*
Note that if *commandB* fails, that will also trigger running *commandC*.

Success and failure are based on the Exit Code of the *command*.
In most cases the Exit Code is the same as the ErrorLevel

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.

```
  command 2> filename       Redirect any error message into a file
  command 2>> filename      Append any error message into a file
 (command)2> filename       Redirect any CMD.exe error into a file
  command > file 2>&1       Redirect errors and output to one file
  command > fileA 2> fileB  Redirect output and errors to separate files

  command 2>&1 >filename    This will fail!
```

```
Redirect to NUL (hide errors)

  command 2> nul            Redirect error messages to NUL
  command >nul 2>&1         Redirect error and output to NUL
  command >filename 2> nul  Redirect output to file but suppress error
 (command)>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.

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

You can also redirect to a printer with > `PRN` or `>LPT1`

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 operator to the front of the line avoids this issue, but is undocumented syntax.

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

Create a new file

Create an empty file using the NUL device:

```
Type NUL >EmptyFile.txt
```
or
```
Copy NUL EmptyFile.txt
```

Multiple commands on one line

In a batch file the default behaviour is to read and expand variables **one line** at a time, if you use `&` to run multiple commands on a single line, then any variable changes will not be visible until execution moves to the next line. For example:

```
 SET /P _cost="Enter the price: " & ECHO %_cost%
```

This behaviour can be changed using `SETLOCAL EnableDelayedExpansion`

Redirect multiple lines

Redirect multiple lines by bracketing a set of commands:

```
(
  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
```

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

`conIN$` and `conOUT$` behave like `stdin` and `stdout`, or 0 and 1 streams but only with internal commands.
SORT - Sort input.
CMD Syntax
TYPE - Display the contents of one or more text files.
Command Redirection - Microsoft Help page (archived)
Successive redirections explained (1>&3) - Stack Overflow.
Equivalent PowerShell: Redirection - Spooling output to a file, piping input.
Equivalent bash command (Linux): Redirection - Spooling output to a file, piping input.

ⓘ