Success and failur
In most cases the E

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.

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 re

```
Set _demo=
>>demofile
```

Create a new

Create an em

```
Type NUL >
```
or
```
Copy NUL E
```
or
```
BREAK > Em
```

Multiple com

In a batch file s on a single line, then any vari

```
 SET /P _c
```

This behavio

Redirect mul

Redirect mul

```
(
  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 a

Although the
All of these c

```
Echo
Echo
Echo>
>C Ec
```

All of them E

If the comma aptures the result of the SORT,

Code like thi

```
Set _
Echo
```

Will inadverte

One solution

```
Echo
```

This assume n 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 conOU
SORT - Sort input.
CMD Syntax
TYPE - Display the
Command Redirec
Successive redire
Equivalent PowerS
Equivalent bash co