# Redirection

Redirection can be used to reassign the **standard input**, **standard output**, and **standard error** devices from their default settings (the keyboard and screen) to another device like the printer or serial port, to a file, or to the clipboard. You must use some discretion when you use redirection with a device; there is no way to get input from the printer, for example.

Redirection always applies to a specific command, and lasts only for the duration of that command. When the command is finished, the assignments for standard input, standard output, and standard error revert to whatever they were before the command.

In the descriptions below, **filename** means either the name of a file or of an appropriate device (**PRN**, **LPT1**, **LPT2**, or **LPT3** for printers; **COM1** to **COM4** for serial ports; **CON** for the keyboard and screen; **CLIP**: for the clipboard; **NUL** for the "null" device, etc.).

Here are the standard redirection options supported by **CMD.EXE** (see below for additional redirection options using numeric file handles):

| | |
|---|---|
| **< filename** | To get input from a file or device instead of from the keyboard |
| **> filename** | Redirect standard output to a file or device |
| **>& filename** | Redirect standard output and standard error to a file or device |
| **>&> filename** | Redirect standard error only to a file or device |

If you want to append output to the end of an existing file, rather than creating a new file, replace the first ">" in the last three commands above with "»" (i.e., use », »&, and »&>).

To use redirection, place the redirection symbol and filename at the end of the command line, after the command name and any parameters. For example, to redirect the output of the **DIR** command to a file called **DIRLIST**, you could use a command line like this:

```
[c:\] dir /b *.dat > dirlist
```

You can use both input and output redirection for the same command, if both are appropriate: For example, this command sends input to **SORT** from the file **DIRLIST**, and sends output from **SORT** to the file **DIRLIST.SRT**:

```
[c:\] sort < dirlist > dirlist.srt
```

You can redirect text to or from the **OS/2** clipboard by using the pseudo-device name **CLIP:** (the colon is required).

If you redirect the output of a single internal command like **DIR**, the redirection ends automatically when that command is done. If you start a batch file with redirection, all of the batch file's output is redirected, and redirection ends when the batch file is done. Similarly, if you use redirection at the end of a command group, all of the output from the command group is redirected, and redirection ends when the command group is done.

When output is directed to a file with >, >&, or >&>, if the file already exists, it will be overwritten. You can protect existing files by using the SETDOS /N1 command, the "Protect redirected output files" setting available on the Options 1 page of the OPTION dialogs, or the NoClobber directive in the .INI

file.

When output is appended to a file with », »&, or »&>, the file will be created if it doesn't already exist. However, if NoClobber is set as described in the above, append redirection will not create a new file; instead, if the output file does not exist a “File not found” or similar error will be displayed.

You can temporarily override the current setting of NoClobber by using an exclamation mark [!] after the redirection symbol. For example, to redirect the output of **DIR** to the file **DIROUT**, and allow overwriting of any existing file despite the NoClobber setting:

```
        [c:\] dir >! dirout
```

Redirection is fully nestable. For example, you can invoke a batch file and redirect all of its output to a file or device. Output redirection on a command within the batch file will take effect for that command only; when the command is completed, output will revert to the redirected output file or device in use for the batch file as a whole.

You can use redirection if you need to create a zero-byte file. To do so, enter **>filename** as a command, with no actual command before the > character.

In addition to the standard redirection options above, **CMD.EXE** also supports the **OS/2 CMD.EXE** syntax:

| | |
|---|---|
| **n>fileRedirect** | handle n to the named file |
| **n>&mRedirect** | handle n to the same place as handle m |

**[n]** and **[m]** are one-digit file handles between 0 and 9. You may not put any spaces between the n and the >, or between the **>**, **&**, and **m** in the second form. **OS/2** interprets “0” as standard input, “1” as standard output, and “2” as standard error. Handles 3 to 9 will probably not be useful unless you have an application which uses those handles for a specific, documented purpose, or you have opened a file with the %**@FILEOPEN** variable function and the file handle is between 3 and 9.

The **n>file** syntax redirects output from handle n to a file. You can use this form to redirect two handles to different places. For example:

```
        [c:\] dir > outfile 2> errfil
```

sends normal output to a file called OUTFILE and any error messages to a file called ERRFILE.

The **n>&m** syntax redirects handle n to the same location as the previously assigned handle m. For example, to send standard error to the same file as standard output, you could use this command:

```
        [c:\] .dir > outfile 2>&1
```

Notice that you can perform the same operations by using standard **CMD.EXE** redirection features. The two examples above could be written as

```
        [c:\] dir > outfile >&> errfile
```

and

```
[c:\] dir >&outfile
```

From:
http://osfree.su/doku/ - **osFree wiki**

Permanent link:
**http://osfree.su/doku/doku.php?id=en:docs:cmd:other:redirection&rev=1400909407**

Last update: **2014/05/24 05:30**