

Bash Redirections Cheat Sheet

| Redirection | Description |
|---|---|
| <code>cmd > file</code> | Redirect the standard output (stdout) of <code>cmd</code> to a file. |
| <code>cmd 1> file</code> | Same as <code>cmd > file</code> . 1 is the default file descriptor (fd) for stdout. |
| <code>cmd 2> file</code> | Redirect the standard error (stderr) of <code>cmd</code> to a file. 2 is the default fd for stderr. |
| <code>cmd >> file</code> | Append stdout of <code>cmd</code> to a file. |
| <code>cmd 2>> file</code> | Append stderr of <code>cmd</code> to a file. |
| <code>cmd &> file</code> | Redirect stdout and stderr of <code>cmd</code> to a file. |
| <code>cmd > file 2>&1</code> | Another way to redirect both stdout and stderr of <code>cmd</code> to a file. This is <u>not</u> the same as <code>cmd 2>&1 > file</code> . <u>Redirection order matters!</u> |
| <code>cmd > /dev/null</code> | Discard stdout of <code>cmd</code> . |
| <code>cmd 2> /dev/null</code> | Discard stderr of <code>cmd</code> . |
| <code>cmd &> /dev/null</code> | Discard stdout and stderr of <code>cmd</code> . |
| <code>cmd < file</code> | Redirect the contents of the file to the standard input (stdin) of <code>cmd</code> . |
| <code>cmd << EOL</code> <code>line1</code> <code>line2</code> <code>EOL</code> | Redirect a bunch of lines to the stdin. If 'EOL' is quoted, text is treated literally. This is called a here-document. |
| <code>cmd <<- EOL</code> <code><tab>foo</code> <code><tab><tab>bar</code> <code>EOL</code> | Redirect a bunch of lines to the stdin and strip the leading tabs. |
| <code>cmd <<< "string"</code> | Redirect a single line of text to the stdin of <code>cmd</code> . This is called a here-string. |
| <code>exec 2> file</code> | Redirect stderr of all commands to a file forever. |
| <code>exec 3< file</code> | Open a file for reading using a custom file descriptor. |
| <code>exec 3> file</code> | Open a file for writing using a custom file descriptor. |
| <code>exec 3<> file</code> | Open a file for reading and writing using a custom file descriptor. |
| <code>exec 3>&-</code> | Close a file descriptor. |
| <code>exec 4>&3</code> | Make file descriptor 4 to be a copy of file descriptor 3. (Copy fd 3 to 4.) |
| <code>exec 4>&3-</code> | Copy file descriptor 3 to 4 and close file descriptor 3. |
| <code>echo "foo" >&3</code> | Write to a custom file descriptor. |
| <code>cat <&3</code> | Read from a custom file descriptor. |
| <code>(cmd1; cmd2) > file</code> | Redirect stdout from multiple commands to a file (using a sub-shell). |
| <code>{ cmd1; cmd2; } > file</code> | Redirect stdout from multiple commands to a file (faster; not using a sub-shell). |
| <code>exec 3<> /dev/tcp/host/port</code> | Open a TCP connection to <code>host:port</code> . (This is a bash feature, not Linux feature). |
| <code>exec 3<> /dev/udp/host/port</code> | Open a UDP connection to <code>host:port</code> . (This is a bash feature, not Linux feature). |
| <code>cmd <(cmd1)</code> | Redirect stdout of <code>cmd1</code> to an anonymous fifo, then pass the fifo to <code>cmd</code> as an argument. Useful when <code>cmd</code> doesn't read from stdin directly. |
| <code>cmd < <(cmd1)</code> | Redirect stdout of <code>cmd1</code> to an anonymous fifo, then redirect the fifo to stdin of <code>cmd</code> . Best example: <code>diff <(find /path1 sort) <(find /path2 sort)</code> . |
| <code>cmd <(cmd1) <(cmd2)</code> | Redirect stdout of <code>cmd1</code> and <code>cmd2</code> to two anonymous fifos, then pass both fifos as arguments to <code>cmd</code> . |
| <code>cmd1 >(cmd2)</code> | Run <code>cmd2</code> with its stdin connected to an anonymous fifo, and pass the filename of the pipe as an argument to <code>cmd1</code> . |
| <code>cmd1 > >(cmd2)</code> | Run <code>cmd2</code> with its stdin connected to an anonymous fifo, then redirect stdout of <code>cmd</code> to this anonymous pipe. |
| <code>cmd1 cmd2</code> | Redirect stdout of <code>cmd1</code> to stdin of <code>cmd2</code> . Pro-tip: This is the same as <code>cmd1 > >(cmd2)</code> , same as <code>cmd2 < <(cmd1)</code> , same as <code>> >(cmd2) cmd1</code> , same as <code>< <(cmd1) cmd2</code> . |
| <code>cmd1 & cmd2</code> | Redirect stdout and stderr of <code>cmd1</code> to stdin of <code>cmd2</code> (bash 4.0+ only). Use <code>cmd1 2>&1 cmd2</code> for older bashes. |
| <code>cmd tee file</code> | Redirect stdout of <code>cmd</code> to a file and print it to screen. |
| <code>exec {filew}>> file</code> | Open a file for writing using a named file descriptor called {filew} (bash 4.1+). |
| <code>cmd 3>&1 1>&2 2>&3</code> | Swap stdout and stderr of <code>cmd</code> . |
| <code>cmd > >(cmd1) 2> >(cmd2)</code> | Send stdout of <code>cmd</code> to <code>cmd1</code> and stderr of <code>cmd</code> to <code>cmd2</code> . |
| <code>cmd1 cmd2 cmd3 cmd4</code> <code>echo \${PIPESTATUS[@]}</code> | Find out the exit codes of all piped commands. |

I explained each one of these redirections in my article All About Bash Redirections:
www.catonmat.net/blog/bash-one-liners-explained-part-three/

Did I miss any redirections? Let me know! Email me peter@catonmat.net, or fork this cheat sheet on github:
www.github.com/pkrumins/bash-redirections-cheat-sheet

A cheat sheet by **Peter Krumins** (peter@catonmat.net, [@pkrumins](https://twitter.com/pkrumins) on twitter)
www.catonmat.net – good coders code, great coders reuse

Released under GNU Free Document License.