Bash Redirections Cheat Sheet

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