# How to Contribute

<!-- vim-markdown-toc GFM -->

\* [Coding Conventions](#coding-conventions)

\* [ShellCheck](#shellcheck)

\* [No no's](#no-nos)

\* [If Statements](#if-statements)

\* [Case Statements](#case-statements)

\* [Making changes to Neofetch](#making-changes-to-neofetch)

\* [Adding support for a new Operating System / Distribution.](#adding-support-for-a-new-operating-system--distribution)

<!-- vim-markdown-toc -->

## Coding Conventions

- Use `bash` built-ins wherever possible.

- Try not to pipe (`|`) at all.

- Limit usage of external commands `$(cmd)`.

- Indent 4 spaces.

- Use [snake\_case](https://en.wikipedia.org/wiki/Snake\_case) for function

and variable names.

- Keep lines below `100` characters long.

- Use `[[ ]]` for tests.

- Quote \*\*EVERYTHING\*\*.

### ShellCheck

For your contribution to be accepted, your changes need to pass

ShellCheck.

```sh

shellcheck neofetch

```

\*\*Note\*\*: If you have trouble installing ShellCheck. You can open a pull

request on the repo and our Travis.ci hook will run ShellCheck for you.

### No no's

- Don’t use GNU conventions in commands.

- Use POSIX arguments and flags.

- Don’t use `cut`.

- Use `bash`'s built-in [parameter expansion](http://wiki.bash-hackers.org/syntax/pe).

- Don’t use `echo`.

- Use `printf "%s\n"`

- Don’t use `bc`.

- Don’t use `sed`.

- Use `bash`'s built-in [parameter expansion](http://wiki.bash-hackers.org/syntax/pe).

- Don’t use `cat`.

- Use `bash`'s built-in syntax (`file="$(< /path/to/file.txt)")`).

- Don’t use `grep "pattern" | awk '{ printf }'`.

- Use `awk '/pattern/ { printf }'`

- Don’t use `wc`.

- Use `${#var}` or `${#arr[@]}`.

### If Statements

If the test only has one command inside of it; use the compact test

syntax. Otherwise the normal `if`/`fi` is just fine.

```sh

# Bad

if [[ "$var" ]]; then

printf "%s\n" "$var"

fi

# Good

[[ "$var" ]] && printf "%s\n" "$var"

# Also good (Use this for longer lines).

[[ "$var" ]] && \

printf "%s\n" "$var"

```

### Case Statements

Case statements need to be formatted in a specific way.

```sh

# Good example (Notice the indentation).

case "$var" in

1) printf "%s\n" 1 ;;

2)

printf "%s\n" "1"

printf "%s\n" "2"

;;

\*)

printf "%s\n" "1"

printf "%s\n" "2"

printf "%s\n" "3"

;;

esac

```

## Making changes to Neofetch

### Adding support for a new Operating System / Distribution.

Adding support for a new OS/Distro requires adding the Name, Logo and

Colors of the OS/Distro to the `get\_distro\_ascii()` function.

The function is located right at the bottom of the script, one function

above `main()`. Inside this function you’ll find an alphabetical list of

each OS/Distro.

Find the spot in the list your new OS/Distro fits into and start

implementing your changes.

If your OS/Distro requires changes to the actual information gathering

functions then you can make these changes in the `get\_\*` functions.

\*\*Syntax\*\*:

- You have to escape back-slashes (`\`). (eg `\\`)

- You can use `${c1}` to `${c6}`to color the ascii.

- These are evaluated \*after\* we read the file.

\*\*Example\*\*:

```sh

"CRUX"\*)

set\_colors 4 5 7 6

read -rd '' ascii\_data <<'EOF'

${c1} odddd

oddxkkkxxdoo

ddcoddxxxdoool

xdclodod olol

xoc xdd olol

xdc ${c2}k00${c1}Okdlol

xxd${c2}kOKKKOkd${c1}ldd

xdco${c2}xOkdlo${c1}dldd

ddc:cl${c2}lll${c1}oooodo

odxxdd${c3}xkO000kx${c1}ooxdo

oxdd${c3}x0NMMMMMMWW0od${c1}kkxo

oooxd${c3}0WMMMMMMMMMW0o${c1}dxkx

docldkXW${c3}MMMMMMMWWN${c1}Odolco

xx${c2}dx${c1}kxxOKN${c3}WMMWN${c1}0xdoxo::c

${c2}xOkkO${c1}0oo${c3}odOW${c2}WW${c1}XkdodOxc:l

${c2}dkkkxkkk${c3}OKX${c2}NNNX0Oxx${c1}xc:cd

${c2} odxxdx${c3}xllod${c2}ddooxx${c1}dc:ldo

${c2} lodd${c1}dolccc${c2}ccox${c1}xoloo

EOF

;;

```