renzok / bash-template

Last active 2 months ago

A template bash script based on google style guide with some little improvements

bash-template

```
#!/bin/bash

# Here short description of this script
# This is just a template to be used for writing new bash scripts

###
# Based on Google Style Guide: https://google.github.io/styleguide/shell.xml
# General remarks
# * Executables should have no extension (strongly preferred) or a .sh extension.
# * Libraries must have a .sh extension and should not be executable
# * SUID and SGID are forbidden on shell scripts.
# * All error messages should go to STDERR.
# * Write todos like this: # TODO(renzok): Handle the unlikely edge cases (bug ####)
# * Indent 2 spaces. No tabs. 80 chars max per line
# * Put ; do and ; then on the same line as the while, for or if.
# * Quoting: https://google.github.io/styleguide/shell.xml#Quoting
# * Function Names: Lower-case, with underscores to separate words.
# ** Separate libraries with ::. Parentheses are required after the function name.
# * prefer shell builtin over separate process
##


##
# Coding tips and tricks:
# http://stackoverflow.com/questions/1167746/how-to-assign-a-heredoc-value-to-a-variabl
#

# Exit immediately if a command exits with a non-zero status.
#
# This might cause problems e.g. using read to read a heredoc cause
```

```
# read to always return non-zero set -o errexit Treat unset variables
# as an error when substituting.
set -o nounset

# 1. section: global constants (all words upper case separated by underscore)
# declare -r CONSTANT_VARIABLE='value'
declare -r TMP_FILE_PREFIX=${TMPDIR:-/tmp}/prog.$$

# as per discussion
# http://stackoverflow.com/questions/4774054/reliable-way-for-a-bash-script-to-get-the-
# but use BASH_SOURCE[0]
declare -r SCRIPTPATH=$( cd $(dirname ${BASH_SOURCE[0]}) > /dev/null; pwd -P )


# 2. section: functions
# Part of a package/library
function mypackage::my_func() {

}

# all progs need to be given as parameters
# e.g. _check_required_programs md5 xsltproc
function _check_required_programs() {

    # Required program(s)
    #req_progs=(readlink date md5)
    for p in ${@}; do
        hash "${p}" 2>&- || \
            { echo >&2 " Required program \"${p}\" not installed or in search PATH.";
              exit 1;
            }
    done

}
function cleanup() {
    rm -f ${TMP_FILE_PREFIX}.*

    echo "always implement this" && exit 100
}


function usage() {
  cat <<EOF

Usage: $0

 TODO
EOF
```

```
}


# Single function
function main() {

  # the optional parameters string starting with ':' for silent errors snd h for help u
  local -r OPTS=':h'

  while builtin getopts ${OPTS} opt "${@}"; do

      case $opt in
          h) usage ; exit 0
             ;;

          \?)
              echo ${opt} ${OPTIND} 'is an invalid option' >&2;
              usage;
              exit ${INVALID_OPTION}
              ;;

          :)
              echo 'required argument not found for option -'${OPTARG} >&2;
              usage;
              exit ${INVALID_OPTION}
              ;;
          *) echo "Too many options. Can not happen actually :)"
             ;;

      esac
  done


  cleanup

  exit 0
}


# Always check return values and give informative return values.
# see https://google.github.io/styleguide/shell.xml#Checking_Return_Values

# set a trap for (calling) cleanup all stuff before process
# termination by SIGHUBs
trap "cleanup; exit 1" 1 2 3 13 15
# this is the main executable function at end of script
main "$@"

```

```
127
128
```

estahn commented on Jun 29, 2018 • edited

```
bin/generate.sh
bin/generate.sh: line 49: syntax error near unexpected token `}'
bin/generate.sh: line 49: `}'
```

If something is added to the function on line 49:

```
bin/generate.sh: line 88: @: unbound variable
```