```
# shellcheck shell=bash
# shellcheck disable=SC2034
#######################################################################################
# Common module for shell scripts.
#
# Written in 2009 by Karel Zimmer <info@karelzimmer.nl>, Creative Commons
# Public Domain Dedication <https://creativecommons.org/publicdomain/zero/1.0>.
#######################################################################################

export TEXTDOMAIN=kz
export TEXTDOMAINDIR=/usr/share/locale

# shellcheck source=/dev/null
source /usr/bin/gettext.sh

#######################################################################################
# Constants
#######################################################################################

declare module_name='kz_common.sh'
declare module_desc
module_desc=$(gettext 'Common module for shell scripts')

declare -i ok=0
declare -i error=1


#######################################################################################
# Variables
#######################################################################################

declare options_short='huv'
declare options_long='help,usage,version'
declare options_usage='[-h|--help] [-u|--usage] [-v|--version]'
declare options_help
options_help="  -h, --help     $(gettext 'give this help list')
  -u, --usage    $(gettext 'give a short usage message')
  -v, --version  $(gettext 'print program version')"

declare -a cmdline_args=()
declare less_options=''
declare logcmd_check=''
declare logcmd=''
declare option_gui=false
# pkexec needs absolute path-name, e.g. ./script -> /path/to/script.
declare program_exec=${0/#./$program_path}
declare text=''
declare title=''
declare usage_line=''

# Terminal attributes, see man terminfo.  Use ${<variabele-name>}.
declare blue=''
declare bold=''
declare green=''
declare normal=''
declare red=''
declare rewrite_line=''
declare yellow=''


#######################################################################################
# Functions
#######################################################################################
```

```
function kz_common.check_dpkgd_snapd {
    local -i dpkg_wait=10

    if find /snap/core/*/var/cache/debconf/config.dat &> /dev/null; then
        # System with snaps.
        while sudo  fuser                                                \
                    /var/{lib/{dpkg,apt/lists},cache/apt/archives}/lock \
                    /var/cache/debconf/config.dat                       \
                    /snap/core/*/var/cache/debconf/config.dat           \
                    &> /dev/null; do
            log "Wait ${dpkg_wait}s for another package manager to finish..."
            sleep $dpkg_wait
        done
    else
        # System without snaps.
        while sudo  fuser                                                \
                    /var/{lib/{dpkg,apt/lists},cache/apt/archives}/lock \
                    /var/cache/debconf/config.dat                       \
                    &> /dev/null; do
            log "Wait ${dpkg_wait}s for another package manager to finish..."
            sleep $dpkg_wait
        done
    fi
}


function kz_common.check_on_ac_power {
    local -i on_battery=0

    on_ac_power |& $logcmd || on_battery=$?
    if [[ on_battery -eq 1 ]]; then
         warning "
$(gettext 'The computer now uses only the battery for power.

It is recommended to connect the computer to the wall socket.')"
        kz_common.wait_for_enter
    fi
}


function kz_common.wait_for_enter {
    if $option_gui; then
        return
    fi
    read -rp "
$(gettext 'Press the Enter key to continue [Enter]: ')" < /dev/tty
}


function kz_common.check_user_root {
    local -i pkexec_rc=0

    if ! kz_common.check_user_sudo; then
        info "$(gettext 'Already performed by the administrator.')"
        exit $ok
    fi
    if [[ $UID -ne 0 ]]; then
        if $option_gui; then
            log "restarted (pkexec $program_exec ${cmdline_args[*]})"
            pkexec "$program_exec" "${cmdline_args[@]}" || pkexec_rc=$?
            exit $pkexec_rc
        else
            log "restarted (exec sudo $program_exec ${cmdline_args[*]})"
            if ! sudo -n true &> /dev/null; then
```

```
                printf  '%s\n' "$(eval_gettext "Authentication is required to \
run \$display_name.")"
            fi
            exec sudo "$program_exec" "${cmdline_args[@]}"
        fi
    fi
}


function kz_common.check_user_sudo {
    # Can user perform sudo?
    if [[ $UID -eq 0 ]]; then
        # For the "grace" period of sudo, or as a root.
        return $ok
    elif groups "$USER" | grep --quiet --regexp='sudo'; then
        return $ok
    else
        return $error
    fi
}


function kz_common.init_script {
    # Script-hardening.
    set -o errtrace
    set -o nounset
    set -o pipefail

    logcmd="systemd-cat --identifier=${program_name:-$module_name}"
    logcmd_check="journalctl --all --boot --identifier=$program_name \
--since='$(date '+%Y-%m-%d %H:%M:%S')'"

    trap 'signal err     $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' ERR
    trap 'signal exit    $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' EXIT
    trap 'signal sighup  $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' SIGHUP  # 1
    trap 'signal sigint  $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' SIGINT  # 2
    trap 'signal sigpipe $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' SIGPIPE #13
    trap 'signal sigterm $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' SIGTERM #15

    log "started ($program_exec $* as $USER)"

    if [[ $(lsb_release --id --short) = 'Debian' && $UID -ne 0 ]]; then
        xhost +si:localuser:root |& $logcmd
    fi

    if [[ -t 1 ]]; then
        set_terminal_attributes
    fi

    cmdline_args=("$@")
    less_options="--LONG-PROMPT --no-init --quit-if-one-screen --quit-on-intr \
--RAW-CONTROL-CHARS --prompt=M$(eval_gettext "Text output \$display_name \
?ltline %lt?L of %L.:byte %bB?s of %s..? .?e (END) :?pB %pB\%. .\
(press h for help or q to quit)")"
    usage_line=$(eval_gettext "Type '\$display_name --usage' for more \
information.")
}


function signal {
    local signal=${1:-unknown}
    local -i lineno=${2:-unknown}
    local function=${3:-unknown}
    local command=${4:-unknown}
```

```
    local -i rc=${5:-$error}
    local rc_desc=''
    local -i rc_desc_signalno=0
    local status="${red}$rc/error${normal}"

    case $rc in
        0)
            rc_desc='successful termination'
            status="${green}$rc/ok${normal}"
            ;;
        1)
            rc_desc='terminated with error'
            ;;
        6[4-9]|7[0-8])                  # 64--78
            rc_desc="open file '/usr/include/sysexits.h' and look for '$rc'"
            ;;
        126)
            rc_desc='command cannot execute'
            ;;
        127)
            rc_desc='command not found'
            ;;
        128)
            rc_desc='invalid argument to exit'
            ;;
        129)                            # SIGHUP (128+1)
            rc_desc='hangup'
            ;;
        130)                            # SIGINT (128+2)
            rc_desc='terminated by control-c'
            ;;
        13[1-9]|140)                    # 140 (128+12)
            rc_desc_signalno=$((rc - 128))
            rc_desc="typ 'trap -l' and look for $rc_desc_signalno"
            ;;
        141)                            # SIGPIPE (128+13)
            rc_desc='broken pipe: write to pipe with no readers'
            ;;
        142)                            # SIGALRM (128+14)
            rc_desc='timer signal from alarm'
            ;;
        143)                            # SIGTERM (128+15)
            rc_desc='termination signal'
            ;;
        14[4-9]|1[5-8][0-9]|19[0-2])    # 144 (128+16)--192 (128+64)
            rc_desc_signalno=$((rc - 128))
            rc_desc="typ 'trap -l' and look for $rc_desc_signalno"
            ;;
        255)
            rc_desc='exit status out of range'
            ;;
        *)
            rc_desc='unknown error'
            ;;
    esac
    log "signal: $signal, line: $lineno, function: $function, command: \
$command, code: $rc ($rc_desc)"

    case $signal in
        err)
             error "
$(eval_gettext "Program \$program_name encountered an error.")"
            exit "$rc"
            ;;
```

```
        exit)
            signal_exit
            log "ended (code=exited, status=$status)"
            trap - ERR EXIT SIGHUP SIGINT SIGPIPE SIGTERM
            exit "$rc"
            ;;
        *)
             error "
$(eval_gettext "Program \$program_name has been interrupted.")"
            exit "$rc"
            ;;
    esac
}


function signal_exit {
    case $program_name in
        kz-install)
            if [[ $rc -ne $ok ]]; then
                log "$(gettext 'If the package manager gives apt errors, \
launch a Terminal window and run:')
[1] ${blue}kz update${normal}
[2] ${blue}sudo update-initramfs -u${normal}"
            fi
            ;;
    esac
}


function set_terminal_attributes {
    blue=$(tput bold; tput setaf 4)
    bold=$(tput bold)
    green=$(tput bold; tput setaf 2)
    normal=$(tput sgr0)
    red=$(tput bold; tput setaf 1)
    rewrite_line=$(tput cuu1; tput el)
    yellow=$(tput bold; tput setaf 3)
}


function kz_common.process_options {
    while true; do
        case $1 in
            -h|--help)
                kz_common.process_option_help
                exit $ok
                ;;
            -u|--usage)
                kz_common.process_option_usage
                exit $ok
                ;;
            -v|--version)
                kz_common.process_option_version
                exit $ok
                ;;
            --)
                break
                ;;
            *)
                shift
                ;;
        esac
    done
}
```

```
function kz_common.process_option_help {
    # shellcheck disable=SC2154
    info "$help

$(eval_gettext "Type 'man \$display_name' for more information.")"
}


function kz_common.process_option_usage {
    # shellcheck disable=SC2154
    info "$usage

$(eval_gettext "Type '\$display_name --help' for more information.")"
}


function kz_common.process_option_version {
    local build_id='1970-01-01'
    local program_year='1970'

    if [[ -e /usr/local/etc/kz-build-id ]]; then
        build_id=$(cat /usr/local/etc/kz-build-id)
    fi
    program_year=$(
        grep '# Written in ' "$program_path/$program_name" |
        cut --delimiter=' ' --fields=4
        ) || true
    if [[ $program_year = '' ]]; then
        program_year='1970'
    fi
    info "$program_name (kz) 365 ($build_id)

$(eval_gettext "Written in \$program_year by Karel Zimmer \
<info@karelzimmer.nl>, Creative Commons
Public Domain Dedication \
<https://creativecommons.org/publicdomain/zero/1.0>.")"
}


function kz_common.reset_terminal_attributes {
    blue=''
    bold=''
    green=''
    normal=''
    red=''
    rewrite_line=''
    yellow=''
}


function log {
    printf '%b\n' "$1" |& $logcmd
}


function info {
    local title=''

    if $option_gui; then
        title=$(eval_gettext "Information \$display_name")
        zenity  --info                  \
                --no-markup             \
```

```
                --width     600         \
                --height    100         \
                --title     "$title"    \
                --text      "$@"        2> >($logcmd) || true
    else
        printf '%b\n' "$@"
    fi
}


function warning {
    local title=''

    if $option_gui; then
        title=$(eval_gettext "Warning \$display_name")
        zenity  --warning               \
                --no-markup             \
                --width     600         \
                --height    100         \
                --title     "$title"    \
                --text      "$@"        2> >($logcmd) || true
    else
        printf "${yellow}%b\n${normal}" "$@" >&2
    fi
}


function error {
    local title=''

    if $option_gui; then
        title=$(eval_gettext "Error message \$display_name")
        zenity  --error                 \
                --no-markup             \
                --width     600         \
                --height    100         \
                --title     "$title"    \
                --text      "$@"        2> >($logcmd) || true
    else
        printf "${red}%b\n${normal}" "$@" >&2
    fi
}
```