```
# shellcheck shell=bash source=/dev/null
################################################################################
# Common module for shell scripts.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2009-2023.
################################################################################

################################################################################
# Import
################################################################################

export TEXTDOMAIN=kz
export TEXTDOMAINDIR=/usr/share/locale
source /usr/bin/gettext.sh


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

module_name='kz_common.sh'
# shellcheck disable=SC2034
module_desc=$(gettext 'Common module for shell scripts')

ok=0
error=1


# shellcheck disable=SC2034
options_short='huv'
# shellcheck disable=SC2034
options_long='help,usage,version'
# shellcheck disable=SC2034
options_usage='[-h|--help] [-u|--usage] [-v|--version]'
# shellcheck disable=SC2034
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=()
logcmd_check=''
logcmd=''
maxrc=0
option_gui=false
# pkexec needs absolute path-name, e.g. ./script -> /path/to/script.
program_exec=${0/#./$program_path}
text=''
title=''

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


################################################################################
# Functions
################################################################################

function check_dpkgd_snapd {
```

```
    local -i dpkg_wait=10
    local text
    text=$(eval_gettext \
            "Wait \${dpkg_wait}s for another package manager to finish...")

    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
            printf '%s\n' "$text"
            sleep $dpkg_wait
        done
    else
        # System without snaps.
        while sudo  fuser                                                \
                    /var/{lib/{dpkg,apt/lists},cache/apt/archives}/lock \
                    /var/cache/debconf/config.dat                       \
                    &> /dev/null; do
            printf '%s\n' "$text"
            sleep $dpkg_wait
        done
    fi
}


function 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.')"
        wait_for_enter
    fi
}


function check_user_root {
    local -i rc=0

    # shellcheck disable=SC2310
    if ! check_user_sudo; then
        info "$(gettext 'Already performed by the administrator.')"
        exit $ok
    fi
    if [[ $UID -ne 0 ]]; then
        if $option_gui; then
            log "restart (pkexec $program_exec ${cmdline_args[*]})"
            pkexec "$program_exec" "${cmdline_args[@]}" || rc=$?
            exit $rc
        else
            log "restart (exec sudo $program_exec ${cmdline_args[*]})"
            exec sudo "$program_exec" "${cmdline_args[@]}"
        fi
    fi
}


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


function info {
    if $option_gui; then
        local title
        title=$(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 init_script {
    # Script-hardening.
    set -o errexit
    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 #3
    trap 'signal sigterm $LINENO ${FUNCNAME:--} "$BASH_COMMAND" $?' SIGTERM #5

    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=("$@")
    # shellcheck disable=SC2034
    usage_line=$(eval_gettext "Type '\$display_name --usage' for more \
information.")
}


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


function maxrc {
    if [[ $rc -gt $maxrc ]]; then
        maxrc=$rc
    fi
}


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


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

$(eval_gettext "Type 'man \$display_name' or see the \
\e]8;;man:\$program_name(1)\e\\\$display_name man page\e]8;;\e\\ for more \
information.")"
}
```

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

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


function process_option_version {
    local build_id='????-??-?? ??:??'
    local grep_expr='# <https://creativecommons.org'
    local program_year='????'

    if [[ -e /usr/local/etc/kz-build-id ]]; then
        build_id=$(cat /usr/local/etc/kz-build-id)
    else
        build_id='????-??-?? ??:??'
    fi

    program_year=$(
        grep    --regexp="$grep_expr" "$program_path/$program_name" |
        cut     --delimiter=' ' --fields=3
        ) || true
    if [[ $program_year = '' ]]; then
        program_year='????'
    fi

    info "$program_name (kz) 365 ($build_id)

$(eval_gettext "Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 \
Universal
<https://creativecommons.org/publicdomain/zero/1.0>, \$program_year")"
}


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


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


function show_log {
    if $option_gui; then
        local temp_log
        temp_log=$(mktemp -t "$program_name-XXXXXXXXXX.log")
        eval "$logcmd_check" > "$temp_log"
        zenity  --text-info                         \
                --width         1300                \
                --height        600                 \
                --title         'Lograpport'        \
```

```
                --filename      "$temp_log"         \
                --cancel-label  "$(gettext 'Exit')" 2> >($logcmd) || true
            rm "$temp_log"
        else
            gnome-terminal --window -- bash -c "$logcmd_check"
        fi
    }
    function signal {
        local signal=${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"
            ;;
        *)
            warning "
$(eval_gettext "Program \$program_name has been interrupted.")"
            exit "$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}"
            else
                # shellcheck disable=SC2154
                rm --force "$cmdsfile" "$simsfile"
            fi
            ;;
        kz-setup)
            if [[ $rc -eq $ok ]]; then
                # shellcheck disable=SC2154
                rm --force "$cmdsfile" "$simsfile"
            fi
            ;;
        *)
            :
            ;;
    esac
}


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


function warning {
    if $option_gui; then
```

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