```
#!/bin/bash
# shellcheck shell=bash source=/dev/null
##############################################################################
# Install kz package.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2016-2023.
##############################################################################

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

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


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

program_name='kz-getdeb'
program_desc=$(gettext 'Install kz package')
display_name=${program_name/kz-/kz }

options_short='huv'
options_long='help,usage,version'
# shellcheck disable=SC2034
options_usage='[-u|--usage] [-h|--help] [-v|--version]'
usage=$(eval_gettext "Usage: \$display_name \$options_usage")
options_help="  -h, --help     $(gettext 'give this help list')
  -u, --usage    $(gettext 'give a short usage message')
  -v, --version  $(gettext 'print program version')"
help="$(eval_gettext "Usage: \$display_name [OPTION...]")

$program_desc.

$options_help"
usage_line=$(eval_gettext "Type '\$display_name --usage' for more \
information.")

bold=$(tput bold)
normal=$(tput sgr0)


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

function init_script {
        set -o errtrace
    set -o nounset
    set -o pipefail
}


function check_input {
    local -i rc=0
    local parsed=''

    parsed=$(
        getopt  --alternative                       \
                --options       "$options_short"    \
```

```
                --longoptions   "$options_long"     \
                --name          "$program_name"     \
                --              "$@"
        )       || rc=$?
    if [[ $rc -ne 0 ]]; then
        printf '%s\n' "$usage_line" >&2
        exit 1
    fi
    eval set -- "$parsed"

    while true; do
        case $1 in
            -h|--help)
                process_option_help
                cleanup
                exit 0
                ;;
            -u|--usage)
                process_option_usage
                cleanup
                exit 0
                ;;
            -v|--version)
                process_option_version
                cleanup
                exit 0
                ;;
            --)
                shift
                break
                ;;
            *)
                shift
                ;;
        esac
    done

    if [[ -n "$*" ]]; then
        printf  "$display_name: $*: %s\n$usage_line\n" \
                "$(gettext 'arguments are not allowed')"
        cleanup
        exit 1
    fi
}


function process_option_help {
    printf '%b\n' "$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 {
    printf '%s\n' "$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='????'
    local program_path
    program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)

    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

    printf '%s\n' "$program_name (kz) 365 ($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 process_input {
    if groups "$USER" | grep --quiet --regexp='sudo'; then
        sudo true
    else
        printf  '%s\n' "$(gettext 'Already performed by the administrator.')"
        term_script
    fi
    install_package
}


function install_package {
    local site_deb='https://karelzimmer.nl/assets/kz/kz_365_all.deb'
    local temp_deb=''

    temp_deb=$(mktemp -t "$program_name-XXXXXXXXXX.deb")
    # Prevent "N: ... user '_apt'. - pkgAcquire::Run (13: Permission denied)".
    chmod o+r "$temp_deb"
    printf "${bold}%s${normal} ($display_name)\n" "$(gettext 'Download')"
    wget --output-document="$temp_deb" $site_deb
    check_dpkgd_snapd
    printf "${bold}%s${normal} ($display_name)\n" "$(gettext 'Install')"
    sudo    DEBIAN_FRONTEND=noninteractive  \
            apt-get                         \
            reinstall                       \
            --yes                           \
            "$temp_deb"
    rm "$temp_deb"
}


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 cleanup {
    # Do not delete kz and kz.1 due to scripts and manuals with the same name.
    rm --force kz.{2..99}
    # But do delete them if in HOME, as described in Checklist installation.
    rm --force "$HOME"/kz "$HOME"kz.1
}


function term_script {
    local title
    title=$(gettext 'Finished')
    local text
    text=$(gettext "Package kz has been successfully installed.

Now follow the steps as described
in the Checklist installation.

Checklist installation can be found
on the site https://karelzimmer.nl,
under Linux.

Type 'exit' to close this window.")

    clear -x
    TERM=ansi whiptail  --backtitle "$display_name - $program_desc" \
                        --title     "$title"                        \
                        --infobox   "$text"                         \
                        16 47

    cleanup
    exit 0
}


################################################################################
# Main
################################################################################

function main {
    init_script
    check_input "$@"
```

```
    process_input
    term_script
}

main "$@"
```