```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Build package kz.
#
# Written in 2021 by Karel Zimmer <info@karelzimmer.nl>, Creative Commons
# Public Domain Dedication <https://creativecommons.org/publicdomain/zero/1.0>.
##############################################################################

set -o errexit
program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)
source "$program_path"/kz_common.sh


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

declare program_name='kz-build'
declare program_desc
program_desc=$(gettext 'Build package kz')
declare display_name=${program_name/kz-/kz }

declare usage
usage=$(eval_gettext "Usage: \$display_name \$options_usage")
declare help
help="$(eval_gettext "Usage: \$display_name [OPTION...]")

$program_desc.

$(gettext 'Options:')
$options_help"

declare docs_repo=$HOME/kz-docs
declare scripts_repo=$HOME/kz-scripts

declare deb_repo=$HOME/kz-deb
declare deb_repo_appdir=$deb_repo/app
declare deb_repo_distdir=$deb_repo/dist
declare deb_name=kz_365_all.deb

declare uploads_repo=$HOME/kz-uploads
declare uploads_repo_appdir=$uploads_repo/app
declare uploads_repo_distdir=$uploads_repo/dist

declare -A uploads_sources=(
            [kz-docs]=$docs_repo
         [kz-scripts]=$scripts_repo
            )
declare -A uploads_targets=(
            [kz-docs]=$uploads_repo_distdir/data/linux/kz-docs
         [kz-scripts]=$uploads_repo_distdir/data/linux/kz-scripts
                [deb]=$uploads_repo_distdir/downloads/kz
            )

declare -A deb_sources=(
                [app]=$deb_repo_appdir
         [kz-scripts]=$scripts_repo
            )
declare -A deb_targets=(
               [dist]=$deb_repo_distdir
              [build]=$deb_repo_distdir/usr/local/etc
             [man_en]=$deb_repo_distdir/usr/share/man/man1
             [man_nl]=$deb_repo_distdir/usr/share/man/nl/man1
```

```
    )


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


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

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

    parsed=$(
        getopt  --alternative                           \
                --options       "$options_short"    \
                --longoptions   "$options_long"     \
                --name          "$display_name"     \
                --              "$@"
        ) || getopt_rc=$?
    if [[ $getopt_rc -ne $ok ]]; then
        info "$usage_line"
        exit $error
    fi
    eval set -- "$parsed"
    kz_common.process_options "$@"

    while true; do
        case $1 in
            --)
                shift
                break
                ;;
            *)
                shift
                ;;
        esac
    done

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


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
    info "* ${bold}Check${normal} ($display_name)"
    check_names
    check_scripts
    info "\n* ${bold}Build${normal} ($display_name)"
    build_uploads
    build_deb
    info "\n* Install${normal} ($display_name)"
```

```
    install_package
}


function check_names {
    local -i check_rc=0
    local app_uploads_repo=$HOME/kz-uploads/app

    info "$(gettext '* Check file names (kz ckname)...')"
    # Call kz-*.
    "$scripts_repo"/usr/bin/kz ckname   "$scripts_repo"     \
                                        "$docs_repo"        \
                                        "$deb_repo"         \
                                        "$app_uploads_repo" || check_rc=$?
    if [[ $check_rc -ne $ok ]]; then
         error "
$(gettext 'Fix all the messages above and then restart the build.')"
        exit $check_rc
    fi
}


function check_scripts {
    local -i check_rc=0

    info "$(gettext '* Check scripts (kz ivp)...')"
    # Call kz-ivp.
    "$scripts_repo"/usr/bin/kz ivp || check_rc=$?
    if [[ $check_rc -ne $ok ]]; then
         error "
$(gettext 'Fix all the messages above and then restart the build.')"
        exit $check_rc
    fi
}


function build_uploads {
    local file=''
    local pdfdir=''
    local tmptxt=''

    info "$(gettext '* Build website...')"

    # Make sure the permissions are correct for the sync.
    chmod 'u=rwx,g=rx,o=rx' -- "$scripts_repo"/*
    chmod 'a-x' -- "$scripts_repo"/*.* "$scripts_repo"/LICENSE

    # Populate kz-uploads/dist/ with modified files in kz-uploads/app/.
    rsync   --archive               \
            --verbose               \
            --checksum              \
            "$uploads_repo_appdir"/ \
            "$uploads_repo_distdir" |& $logcmd

    # Populate kz-uploads/dist/ with modified files in repo kz-docs, and create
    # a PDF of these modified files.
    cd "${uploads_sources[kz-docs]}"
    for file in *.odt *.txt; do
        if ! diff "$file" "${uploads_targets[kz-docs]}/$file" |& $logcmd; then
            cp  --preserve                      \
                --verbose                       \
                "$file"                         \
                "${uploads_targets[kz-docs]}"   |& $logcmd
            # Craete a PDF.
```

```
            lowriter    --headless                                  \
                        --convert-to pdf                            \
                        --outdir "${uploads_targets[kz-docs]}"  \
                        "$file"                                     |& $logcmd
        fi
    done
    cd "$HOME"

    # Fill kz-uploads/dist/ with modified files in repo kz-scripts, and create
    # a PDF of these modified files.
    cd "${uploads_sources[kz-scripts]}"
    while read -r file; do
        if ! diff "$file" "${uploads_targets[kz-scripts]}/$file" |& $logcmd
        then
            cp  --parents                           \
                --preserve                          \
                --verbose                           \
                "$file"                             \
                "${uploads_targets[kz-scripts]}"    |& $logcmd
            # Craete a PDF.
            if grep --quiet --regexp='^''.TH ' "$file"; then
                # Man page file.
                man --troff "${uploads_sources[kz-scripts]}/$file"  |
                ps2pdf - "${uploads_targets[kz-scripts]}/$file.pdf" |& $logcmd
            else
                # Script file.

                # Must copy each file with suffix e.g. .txt added (tmptxt)
                # before converting because:
                # 1. desktop-files have XML inside which gets interpreted by
                #    Libre Office,
                # 2. 'lowriter convert-to pdf' replaces last suffix (if any) by
                #    .pdf.
                tmptxt=/tmp/$(basename "$file").txt
                cp "$file" "$tmptxt"
                pdfdir=${uploads_targets[kz-scripts]}/$(dirname "$file")

                lowriter    --headless          \
                            --convert-to pdf    \
                            --outdir "$pdfdir"  \
                            "$tmptxt"           |& $logcmd
                rm "$tmptxt"
            fi
        fi
    done < <(
            find .                      \
            -type f                     \
            -not -path './.git*'        \
            -not -path '*/__pycache__*' \
            -not -name kz.mo            \
            -not -name LICENSE          \
            -not -name README.md        \
            -print
            )
    cd "$HOME"
}


function build_deb {
    local file=''

    info "$(gettext '* Build package...')"

    # Fill kz-deb/dist/ with kz-deb/app/.
```

```
    rsync   --archive               \
            --delete                \
            --verbose               \
            --exclude='README.md'   \
            --exclude='.git*'       \
            --delete-excluded       \
            "${deb_sources[app]}"/  \
            "${deb_targets[dist]}"  |& $logcmd

    # Fill kz-deb/dist/ with repo kz-scripts.
    rsync   --archive                       \
            --verbose                       \
            --exclude='__pycache__'         \
            --exclude='LICENSE'             \
            --exclude='README.md'           \
            --exclude='.git*'               \
            --exclude='kz.po'               \
            --exclude='kz.pot'              \
            "${deb_sources[kz-scripts]}"/   \
            "${deb_targets[dist]}"          |& $logcmd

    # Compress man pages.
    gzip    --best                      \
            --force                     \
            "${deb_targets[man_en]}"/*  \
            "${deb_targets[man_nl]}"/*  |& $logcmd

    # Capture build id Debian package.
    printf '%s' "$(date +%Y-%m-%d)" > "${deb_targets[build]}/$program_name"-id

    # Create Debian package in kz-uploads.
    # Debian supports xz as maximum compression, the default Ubuntu is zst.
    fakeroot    dpkg-deb                                \
                --build                                 \
                -Zxz                                    \
                "$deb_repo_distdir"                     \
                "${uploads_targets[deb]}/$deb_name" |& $logcmd
}


function install_package {
    info "$(gettext '* Install package...')"
    sudo    DEBIAN_FRONTEND=noninteractive      \
            apt-get                             \
            reinstall                           \
            --yes                               \
            "${uploads_targets[deb]}/$deb_name" |& $logcmd
}


function term_script {
    exit $ok
}


################################################################################
# Script
################################################################################

function main {
    kz_common.init_script "$@"
    check_input "$@"
    process_input
    term_script
```

```
}

main "$@"
```