```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Build package kz.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2021-2023.
##############################################################################

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

program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)
source "$program_path"/kz_common.sh 2> >(systemd-cat) || exit 1


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

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

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

$program_desc.

$(gettext 'Options:')
$options_help"

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

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

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

declare -A uploads_sources=(
            [kz-docs]=$docs_repo
         [kz-scripts]=$scripts_repo
            )
declare -A uploads_targets=(
            [kz-docs]=$uploads_repo_distdir/assets/kz-docs
         [kz-scripts]=$uploads_repo_distdir/assets/kz-scripts
                [deb]=$uploads_repo_distdir/assets/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
```

```
    )


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

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

    parsed=$(
        getopt  --alternative                           \
                --options       "$options_short"    \
                --longoptions   "$options_long"     \
                --name          "$display_name"     \
                --              "$@"
        ) || rc=$?
    if [[ $rc -ne $ok ]]; then
        printf '%s\n' "$usage_line"
        exit $error
    fi
    eval set -- "$parsed"
    process_option "$@"

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

    if [[ -n "$*" ]]; then
        printf  "$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
    printf "${bold}%s${normal} ($display_name)\n" "$(gettext 'Check - pre')"
    check_repos
    check_scripts
    printf "\n${bold}%s${normal} ($display_name)\n" "$(gettext 'Build')"
    build_uploads
    generate_mo
    build_deb
    printf "\n${bold}%s${normal} ($display_name)\n" "$(gettext 'Install')"
    install_package
    printf "\n${bold}%s${normal} ($display_name)\n" "$(gettext 'Check - post')"
    run_ivp
    check_website
```

```
}


function check_repos {
    info "$(gettext 'Check that all repos are on branch main...')"
    for repo in $deb_repo $docs_repo $scripts_repo $uploads_repo; do
        cd "$repo"
        if [[ $(git branch --show-current) != 'main' ]]; then
            warning "$(eval_gettext "Repo \$repo not on branch main.")"
        fi
    done
    info "$(gettext 'Check that all repos are clean...')"
    for repo in $deb_repo $docs_repo $scripts_repo $uploads_repo; do
        cd "$repo"
        if ! git diff-index --quiet HEAD; then
            warning "$(eval_gettext "Repo \$repo is not clean.")"
            git status
        fi
    done
}


function check_scripts {
    local -i rc=0

    info "$(gettext 'Check scripts (kz check)...')"
    # Call kz-check.
    "$scripts_repo"/usr/bin/kz check || rc=$?
    if [[ $rc -ne $ok ]]; then
        error "
$(gettext 'Fix all the messages above.')"
        exit $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' -- "$scripts_repo"/usr/bin/*
    chmod 'a-x' -- "$scripts_repo"/usr/bin/*.*

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

    # Make sure all necessary directories are available for the sync with the
    # cp command.
    for dir in "${uploads_targets[@]}"; do
        mkdir --parents "$dir" |& $logcmd
    done

    # 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]}"  \
                        "$file"
        fi
    done
    cd "$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"
            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"
                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 generate_mo {
```

```
    local lc_messages=/home/"$USER"/kz-scripts/usr/share/locale/nl/LC_MESSAGES

    info "$(gettext 'Generate .mo file...')"
    msgfmt --output-file="$lc_messages"/kz.mo  "$lc_messages"/kz.po
}


function build_deb {
    local build_id=''
    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.
    build_id=$(date '+%Y-%m-%d %H:%M')
    printf '%s' "$build_id" > "${deb_targets[build]}/$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 run_ivp {
    local -i rc=0

    info "$(gettext 'Run IVP (kz ivp)...')"
    # Call kz-ivp.
    kz ivp || rc=$?
    if [[ $rc -ne $ok ]]; then
        error "
$(gettext 'Fix all the messages above.')"
        exit $rc
    fi
}


function check_website {
    local -i rc=0

    info "$(gettext 'Check website (rchecklink)...')"
    # Call rchecklink.
    rchecklink || rc=$?
    if [[ $rc -ne $ok ]]; then
        error "
$(gettext 'Fix all the messages above.')"
        exit $rc
    fi
}


function term_script {
    exit $ok
}


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

function main {
    init_script "$@"
    check_input "$@"
    process_input
    term_script
}

main "$@"
```