```
#!/bin/bash
# shellcheck source=kz-common.sh
################################################################################
# Bestandsnamen controleren.
#
# Geschreven door Karel Zimmer <info@karelzimmer.nl>.
################################################################################
PROGRAM_PATH=$(realpath "$(dirname  "$0")")
source "$PROGRAM_PATH"/kz-common.sh
PROGRAM_NAME=kz-ckname
DISPLAY_NAME=${PROGRAM_NAME/kz-/kz }
RELEASE_YEAR=2012

VERSION_NUMBER=13.00.08
VERSION_DATE=2021-10-04


################################################################################
# Global constants
################################################################################

readonly    RUN_AS_SUPERUSER=false
readonly    OPTIONS_SHORT=$OPTIONS_SHORT_COMMON
readonly    OPTIONS_LONG=$OPTIONS_LONG_COMMON
readonly    USAGE="Gebruik: $DISPLAY_NAME $OPTIONS_USAGE_COMMON
                   [MAP...]"
readonly    HELP="Gebruik: $DISPLAY_NAME [OPTIE...] [MAP...]

Bestandsnamen controleren.

Opties:
$OPTIONS_HELP_COMMON

Argumenten:
  MAP           begin te controleren vanaf map MAP"

declare -ir FILENAME_MAXLEN=142
readonly    START_DFLT=$HOME


################################################################################
# Global variables
################################################################################

declare     ARGUMENT_MAP=false
declare -a  MAP_ARGUMENT=()


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

check_input() {
    local   -i  map_arg_num=0
    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 $SUCCESS ]]; then
        printf '%s\n' "$USAGELINE" >&2
        exit $ERROR
    fi
```

```
    eval set -- "$parsed"
    process_common_options "$@"

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

    while [[ "$*" ]]; do
        ARGUMENT_MAP=true
        MAP_ARGUMENT[$map_arg_num]=$1
        ((++map_arg_num))
        shift
    done
    if ! $ARGUMENT_MAP; then
        MAP_ARGUMENT[0]=$START_DFLT
    fi
    for dir in ${MAP_ARGUMENT[*]}; do
        if ! [[ -d $dir ]]; then
                TEXT="map '$dir' bestaat niet"
                printf  "$DISPLAY_NAME: %s\n%s\n"   \
                        "$TEXT"                     \
                        "$USAGELINE"                >&2
            exit $ERROR
        fi
    done

    # Een non-gui script gestart met optie gui.
    if $OPTION_GUI; then
        OPTION_GUI=false
        DESKTOP_TERMINAL=false
    fi

    check_user
}


process_input() {
    check_files_and_folders
}


check_files_and_folders() {
    local       dirname
    local       file
    local       basename
    local       good
    local   -i  count=0

    # Het maakt niet uit of in de bestandsnaam speciale tekens voorkomen zoals
    # tab, spatie, enz.  Hiervoor zorgt de find met print0,en de read met IFS=
    # en als delimiter de null character die niet mag voorkomen in een
    # bestandsnaam.  N.B.: In Linux is alles een bestand!
    while IFS= read -r -d $'\0' file; do

        dirname=$(dirname "$file")
        basename=$(basename "$file")

        # Verwijder de slechte tekens, \ is escape voor " en :
        good=$(printf '%s'"$file" | tr --delete '?"\\<>*|:')

```

```
  137:         if ! [[ $file = "$good" ]]; then
  138:             ((++count))
  139:             if [[ -d "$file" ]]; then
  140:                 info "BadName  DIR '$file'."
  141:             elif [[ -f "$file" ]]; then
  142:                 info "BadName FILE '$basename' in map '$dirname'."
  143:             else
  144:                 info "BadName SYML '$file'."
  145:             fi
  146:         fi
  147:
  148:         if [[ ${#basename} -gt $FILENAME_MAXLEN ]]; then
  149:             ((++count))
  150:             if [[ -d "$file" ]]; then
  151:                 info "BadLen.  DIR '$basename'."
  152:             else
  153:                 info "BadLen. FILE '$basename' in map '$dirname'."
  154:             fi
  155:         fi
  156:
  157:     done < <(
  158:         find    "${MAP_ARGUMENT[@]}"    \
  159:                 -type f                 \
  160:                 -print0                 \
  161:                 -or                     \
  162:                 -type d                 \
  163:                 -print0                 \
  164:                 -or                     \
  165:                 -type l                 \
  166:                 -print0
  167:         )
  168:
  169:     if [[ $count -eq 0 ]]; then
  170:         return $SUCCESS
  171:     elif [[ $count -eq 1 ]]; then
  172:         warning 'Er is een fout gevonden.'
  173:     else
  174:         warning "Er zijn $count fouten gevonden."
  175:     fi
  176: }
  177:
  178:
  179: term_script() {
  180:     exit $SUCCESS
  181: }
  182:
  183:
  184: ###############################################################################
  185: # Main line
  186: ###############################################################################
  187:
  188: main() {
  189:     init_script "$@"
  190:     check_input "$@"
  191:     process_input
  192:     term_script
  193: }
  194:
  195:
  196: main "$@"
  197:
  198:
  199: # EOF
```