```
#!/bin/bash
# shellcheck source=kzcommon.sh
# ------------------------------------------------------------------------------
# Bestandsnamen controleren.
#
# Geschreven door Karel Zimmer <info@karelzimmer.nl>.
#
# Auteursrecht (c) 2012-2021 Karel Zimmer.
# GNU Algemene Publieke Licentie <https://www.gnu.org/licenses/gpl.html>.
#
# RelNum=12.03.00
# RelDat=2021-01-18
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# Global constants
# ------------------------------------------------------------------------------
source "$(dirname "$0")"/kzcommon.sh
declare -ir FILENAME_MAXLEN=142
readonly START_DFLT=$HOME

# Bij aanpassingen ook .completion aanpassen!
readonly OPTIONS_SHORT=$OPTIONS_SHORT_COMMON
readonly OPTIONS_LONG=$OPTIONS_LONG_COMMON
readonly USAGE="Gebruik: $PROGNAME $OPTIONS_USAGE_COMMON [MAP...]"
readonly HELP="Gebruik: $PROGNAME [OPTIE...] [--] [MAP...]

Bestandsnamen controleren.

Opties:
$OPTIONS_HELP_COMMON

Argumenten:
  MAP           begin te controleren vanaf map MAP"

# ------------------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------------------
declare ARGUMENT_MAP=false
declare -a MAP_ARGUMENT=''

# ------------------------------------------------------------------------------
# Functions
# ------------------------------------------------------------------------------
check_input() {
    local -i getopt_rc=0
    local -i map_arg_num=0
    local parsed=''

    parsed=$(
        getopt  --alternative                       \
                --options       "$OPTIONS_SHORT"    \
                --longoptions   "$OPTIONS_LONG"     \
                --name          "$PROGNAME"         \
                -- "$@"
        )       || getopt_rc=$?
    if [[ $getopt_rc -ne 0 ]]; then
        printf '%s\n' "$USAGELINE" >&2
        exit $ERROR
    fi
    eval set -- "$parsed"
    process_general_options "$@"

    while true; do
        case $1 in
            --)
                shift
                break
```

```
   69:                 ;;
   70:             *)
   71:                 shift
   72:                 ;;
   73:         esac
   74:     done
   75:
   76:     while [[ "$*" ]]; do
   77:         ARGUMENT_MAP=true
   78:         MAP_ARGUMENT[$map_arg_num]=$1
   79:         ((++map_arg_num))
   80:         shift
   81:     done
   82:
   83:     # Een non-gui script gestart met optie gui.
   84:     if $OPTION_GUI; then
   85:         OPTION_GUI=false
   86:         TERMINAL=true
   87:     fi
   88:
   89:     check_user
   90: }
   91:
   92: process_input() {
   93:     if ! $ARGUMENT_MAP; then
   94:         MAP_ARGUMENT[0]=$START_DFLT
   95:     fi
   96:
   97:     for dir in ${MAP_ARGUMENT[*]}; do
   98:         if ! [[ -d $dir ]]; then
   99:             error "Map '$dir' bestaat niet."
  100:             exit $ERROR
  101:         fi
  102:     done
  103:
  104:     check_files_and_folders
  105: }
  106:
  107: check_files_and_folders() {
  108:     local dirname
  109:     local file
  110:     local basename
  111:     local good
  112:     local -i count=0
  113:
  114:     # Het maakt niet uit of in de bestandsnaam speciale tekens voorkomen zoals
  115:     # tab, spatie, enz.  Hiervoor zorgt de find met print0,en de read met IFS=
  116:     # en als delimiter de null character die niet mag voorkomen in een
  117:     # bestandsnaam.  N.B.: In Linux is alles een bestand!
  118:     while IFS= read -r -d $'\0' file; do
  119:
  120:         dirname=$(dirname "$file")
  121:         basename=$(basename "$file")
  122:
  123:         # Verwijder de slechte tekens, \ is escape voor " en :
  124:         good=$(printf '%s'"$file" | tr --delete '?"\\<>*|:')
  125:
  126:         if ! [[ $file = "$good" ]]; then
  127:             ((++count))
  128:             if [[ -d "$file" ]]; then
  129:                 info "BadName  DIR '$file'."
  130:             elif [[ -f "$file" ]]; then
  131:                 info "BadName FILE '$basename' in map '$dirname'."
  132:             else
  133:                 info "BadName SYML '$file'."
  134:             fi
  135:         fi
  136:
```

```
  137:         if [[ ${#basename} -gt $FILENAME_MAXLEN ]]; then
  138:             ((++count))
  139:             if [[ -d "$file" ]]; then
  140:                 info "BadLen.  DIR '$basename'."
  141:             else
  142:                 info "BadLen. FILE '$basename' in map '$dirname'."
  143:             fi
  144:         fi
  145:
  146:     done < <(
  147:         find    "${MAP_ARGUMENT[@]}"    \
  148:                 -type f                 \
  149:                 -print0                 \
  150:                 -or                     \
  151:                 -type d                 \
  152:                 -print0                 \
  153:                 -or                     \
  154:                 -type l                 \
  155:                 -print0
  156:         )
  157:
  158:     if [[ $count -eq 0 ]]; then
  159:         return $SUCCESS
  160:     elif [[ $count -eq 1 ]]; then
  161:         warning 'Er is Ã\203Â©Ã\203Â©n fout gevonden.'
  162:     else
  163:         warning "Er zijn $count fouten gevonden."
  164:     fi
  165: }
  166:
  167: term_script() {
  168:     exit $SUCCESS
  169: }
  170:
  171: # ----------------------------------------------------------------------------
  172: # Main line
  173: # ----------------------------------------------------------------------------
  174: main() {
  175:     init_script
  176:     check_input "$@"
  177:     process_input
  178:     term_script
  179: }
  180:
  181: main "$@"
  182:
  183: # EOF
```