```
#!/bin/bash
# shellcheck source=kzcommon.sh
# ------------------------------------------------------------------------------
# USB-sticks maken.
#
# Geschreven door Karel Zimmer <info@karelzimmer.nl>.
#
# Auteursrecht (c) 2014-2021 Karel Zimmer.
# GNU Algemene Publieke Licentie <https://www.gnu.org/licenses/gpl.html>.
#
# RelNum=17.04.00
# RelDat=2021-01-18
# ------------------------------------------------------------------------------

# ------------------------------------------------------------------------------
# Global constants
# ------------------------------------------------------------------------------
source "$(dirname "$0")"/kzcommon.sh
readonly DEVICE_DEFAULT=/dev/sdb
readonly SOURCE_DEFAULT=/home/${SUDO_USER:-$USER}/Downloads
readonly RUN_AS_SUPERUSER=true

# Bij aanpassingen ook .completion aanpassen!
readonly OPTIONS_SHORT=$OPTIONS_SHORT_COMMON’s:t:’
readonly OPTIONS_LONG=$OPTIONS_LONG_COMMON’source:,target:’
readonly USAGE="Gebruik: $PROGNAME [-s|--source=BRON] [-t|--target=DOEL]
                 $OPTIONS_USAGE_COMMON MAP"
readonly HELP="Gebruik: $PROGNAME [OPTIE...] [--] MAP

USB-sticks maken.

Apties:
  -s --source=BRONMAP
                gebruik beeldbestanden in de BRONMAP
  -t --target=APPARAAT
                gebruik de opgegeven USB-stick-device
$OPTIONS_HELP_COMMON

Argumenten:
  MAP           gebruik beelbestanden (.iso) in opgegeven map (verplicht)"

# ------------------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------------------
declare ARGUMENT_SOURCE=false
declare DEVICE=’’
declare NO_FILES_FOUND=true
declare OPTION_SOURCE=false
declare OPTION_TARGET=false
declare SOURCEDIR=’’
declare SOURCE_ARGUMENT=’’
declare TARGET_ARGUMENT=’’

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

    parsed=$(
        getopt  --alternative                       \
                --options       "$OPTIONS_SHORT"    \
                --longoptions   "$OPTIONS_LONG"     \
                --name          "$PROGNAME"         \
                -- "$@"
        )       || getopt_rc=$?
    if [[ $getopt_rc -ne 0 ]]; then
```

```
   69:         printf '%s\n' "$USAGELINE" >&2
   70:         exit $ERROR
   71:     fi
   72:     eval set -- "$parsed"
   73:     process_general_options "$@"
   74:
   75:     while true; do
   76:         case $1 in
   77:             -s|--source)
   78:                 if $OPTION_SOURCE; then
   79:                     printf "$PROGNAME: %s\n%s\n" "optie '$1' Ã©Ã©nmaal opgeven" \
   80: "$USAGELINE" >&2
   81:                     exit $ERROR
   82:                 else
   83:                     OPTION_SOURCE=true
   84:                     SOURCE_ARGUMENT=$2
   85:                 fi
   86:                 shift 2
   87:                 ;;
   88:             -t|--target)
   89:                 if $OPTION_TARGET; then
   90:                     printf "$PROGNAME: %s\n%s\n" "optie '$1' Ã©Ã©nmaal opgeven" \
   91: "$USAGELINE" >&2
   92:                     exit $ERROR
   93:                 else
   94:                     OPTION_TARGET=true
   95:                     TARGET_ARGUMENT=$2
   96:                 fi
   97:                 shift 2
   98:                 ;;
   99:             --)
  100:                 shift
  101:                 break
  102:                 ;;
  103:             *)
  104:                 shift
  105:                 ;;
  106:         esac
  107:     done
  108:
  109:     while [[ "$*" ]]; do
  110:         if $ARGUMENT_SOURCE; then
  111:             printf "$PROGNAME: %s\n%s\n" "argument 'MAP' Ã©Ã©nmaal opgeven" \
  112: "$USAGELINE" >&2
  113:             exit $ERROR
  114:         else
  115:             ARGUMENT_SOURCE=true
  116:             SOURCE_ARGUMENT=$1
  117:         fi
  118:         shift
  119:     done
  120:
  121:     if $OPTION_SOURCE; then
  122:         SOURCEDIR=$SOURCE_ARGUMENT
  123:     else
  124:         SOURCEDIR=$SOURCE_DEFAULT
  125:     fi
  126:     if ! [[ -d $SOURCEDIR ]]; then
  127:         printf "$PROGNAME: %s\n%s\n" "map '$SOURCEDIR' bestaat niet" \
  128: "$USAGELINE" >&2
  129:         exit $ERROR
  130:     fi
  131:
  132:     if $OPTION_TARGET; then
  133:         DEVICE=$TARGET_ARGUMENT
  134:     else
  135:         DEVICE=$DEVICE_DEFAULT
  136:     fi
```

```
137:
138:     # Een non-gui script gestart met optie gui.
139:     if $OPTION_GUI; then
140:         OPTION_GUI=false
141:         TERMINAL=true
142:     fi
143:
144:     check_user
145:     request_input
146: }
147:
148: request_input() {
149:     local msg='Gebruik optie --target om een ander device op te geven.'
150:
151:     warning 'De volgende /dev/sd schijven en partities zijn aanwezig:'
152:     if  findmnt --list                  \
153:                 --output=SOURCE,TARGET  |
154:         grep    --regexp='/dev/sd'; then
155:         warning "
156: $DEVICE wordt gebruikt als doel en OVERSCHREVEN!"
157:         while true; do
158:             read -rp 'Is dit OK? [j/N]: ' < /dev/tty
159:             case $REPLY in
160:                 j*|J*)
161:                     break
162:                     ;;
163:                 n*|N*|'')
164:                     printf '%s\n%s\n%s\n' "$msg" "$USAGELINE" 'Gestopt.'
165:                     exit $SUCCESS
166:                     ;;
167:                 *)
168:                     printf '%b\n' "${UP_ONE_LINE}${ERASE_LINE}"
169:                     continue
170:                     ;;
171:             esac
172:         done
173:     else
174:         warning "
175: Geen $DEVICE gevonden om te gebruiken als doel!"
176:         printf '%s\n%s\n%s\n' "$msg" "$USAGELINE" 'Gestopt.'
177:         exit $WARNING
178:     fi
179: }
180:
181: process_input() {
182:     local -i isosize=0
183:     local usbname=''
184:     local file=''
185:
186:     NO_FILES_FOUND=true
187:
188:     for file in "$SOURCEDIR"/*.iso; do
189:
190:         if ! [[ -f "$file" ]]; then
191:             continue
192:         fi
193:
194:         NO_FILES_FOUND=false
195:         usbname=$(basename "$file" .iso | tr '-' ' ')
196:
197:         printf "\n${BOLD}%s${NORMAL}\n" "$usbname"
198:
199:         while true; do
200:             read -rp 'maken/Overslaan/stoppen? [m/O/s]: ' < /dev/tty
201:             case $REPLY in
202:                 m*|M*)
203:                     isosize=$(
204:                         du  --apparent-size \
```

```
205:                             --block-size=1  \
206:                             "$file"         |
207:                         awk '{print $1}'
208:                         )
209:                     check_on_ac_power
210:                     create_usb_stick "$file"
211:                     while true; do
212:                         read -rp "Nog een USB-stick '$usbname' maken? \
213: [j/N]: " < /dev/tty
214:                         case $REPLY in
215:                             j*|J*)
216:                                 check_on_ac_power
217:                                 create_usb_stick "$file"
218:                                 continue
219:                                 ;;
220:                             n*|N*|'')
221:                                 printf '%s\n' 'Klaar.'
222:                                 break
223:                                 ;;
224:                             *)
225:                                 printf '%b\n' "${UP_ONE_LINE}${ERASE_LINE}"
226:                                 continue
227:                                 ;;
228:                         esac
229:                     done
230:                     break
231:                     ;;
232:                 o*|O*|'')
233:                     printf '%s\n' 'Overgeslagen.'
234:                     break
235:                     ;;
236:                 s*|S*)
237:                     printf '%s\n' 'Gestopt.'
238:                     exit $SUCCESS
239:                     ;;
240:                 *)
241:                     printf '%b\n' "${UP_ONE_LINE}${ERASE_LINE}"
242:                     continue
243:                     ;;
244:             esac
245:         done
246:     done
247:
248:     if $NO_FILES_FOUND; then
249:         warning "Geen beeldbestanden (.iso) gevonden in map '$SOURCEDIR'."
250:         exit $WARNING
251:     fi
252: }
253:
254: create_usb_stick() {
255:     local dd_from=${1:-dd_from?}
256:
257:     read -rp "Plaats USB-stick '$usbname', gevolgd door Enter [Enter]: " < \
258: /dev/tty
259:     if [[ -e $DEVICE ]]; then
260:         info "USB-stick $usbname wordt gemaakt ..."
261:
262:         dd  if="$dd_from"       \
263:             status=none         |
264:         pv  --size="$isosize"   |
265:         dd  of=$DEVICE          \
266:             bs=4M               \
267:             status=none
268:
269:         info "Gegevens schrijven naar $usbname (dit kan even duren) ..."
270:         sync
271:
272:         info "USB-stick '$usbname' is gemaakt."
```

```
  273:
  274:         info "Ter controle is er een Kernel-based Virtuele Machine (KVM) \
  275: gestart."
  276:         if ! qemu-system-x86_64 -enable-kvm     \
  277:                                 -m 2048         \
  278:                                 -hda $DEVICE    |& $LOGCMD; then
  279:             true
  280:         fi
  281:
  282:         if ! umount "$DEVICE" 2> /dev/null; then
  283:             read -rp "Ontkoppel de USB-stick '$usbname', gevolgd door Enter \
  284: [Enter]: " < /dev/tty
  285:         fi
  286:         read -rp "Verwijder de USB-stick '$usbname', gevolgd door Enter \
  287: [Enter]: " < /dev/tty
  288:     else
  289:         warning "USB-stick '$usbname' is niet aanwezig op $DEVICE."
  290:         exit $WARNING
  291:     fi
  292: }
  293:
  294: term_script() {
  295:     exit $SUCCESS
  296: }
  297:
  298: # ----------------------------------------------------------------------------
  299: # Main line
  300: # ----------------------------------------------------------------------------
  301: main() {
  302:     init_script
  303:     check_input "$@"
  304:     process_input
  305:     term_script
  306: }
  307:
  308:
  309: main "$@"
  310:
  311: # EOF
```