```
#!/bin/bash
# shellcheck source=kz-common.sh
################################################################################
# Scripts controleren.
#
# Geschreven door Karel Zimmer <info@karelzimmer.nl>.
################################################################################
PROGRAM_PATH=$(realpath "$(dirname  "$0")")
source "$PROGRAM_PATH"/kz-common.sh
PROGRAM_NAME=kz-ivp
DISPLAY_NAME=${PROGRAM_NAME/kz-/kz }
RELEASE_YEAR=2015

VERSION_NUMBER=14.04.02
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"
readonly HELP="Gebruik: $DISPLAY_NAME [OPTIE...]

Scripts controleren.

Opties:
$OPTIONS_HELP_COMMON"


################################################################################
# Globale variabelen
################################################################################

declare -i MAXRC=0


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

check_input() {
    local   -i  getopt_rc=0
    local       parsed=''
    local       prompt=''

    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
```

```
    69:                 ;;
    70:             *)
    71:                 shift
    72:                 ;;
    73:         esac
    74:     done
    75:
    76:     if [[ "$*" ]]; then
    77:         TEXT='geen argumenten opgeven'
    78:         printf "$DISPLAY_NAME: %s\n%s\n" "$TEXT" "$USAGELINE" >&2
    79:         exit $ERROR
    80:     fi
    81:
    82:     # Een non-gui script gestart met optie gui.
    83:     if $OPTION_GUI; then
    84:         OPTION_GUI=false
    85:         DESKTOP_TERMINAL=false
    86:     fi
    87:
    88:     check_user
    89:
    90:     # Regel sudo maar gelijk voor verderop.
    91:     prompt="Authenticatie is vereist om delen van $DISPLAY_NAME uit te voeren.
    92: [sudo] wachtwoord voor %p: "
    93:     sudo --prompt="$prompt" true
    94:
    95:     check_dependencies
    96:
    97: }
    98:
    99:
   100: check_dependencies() {
   101:     if ! command -v pycodestyle 1> /dev/null; then
   102:         info 'Installeer pycodestyle...'
   103:         sudo apt-get install --yes pycodestyle |& $LOGCMD
   104:     fi
   105:     if ! command -v shellcheck 1> /dev/null; then
   106:         info 'Installeer shellcheck...'
   107:         # Debian's pakket is oud, snap is nieuwer.
   108:         sudo snap install shellcheck |& $LOGCMD
   109:     fi
   110: }
   111:
   112:
   113: process_input() {
   114:     local script=''
   115:     local scriptsdir=$HOME/kz-scripts
   116:
   117:     cd "$scriptsdir" || exit $ERROR
   118:     for script in *; do
   119:         if ! [[ -f $script ]]; then
   120:             continue
   121:         fi
   122:         case $script in
   123:             LICENSE|README.md)
   124:                 continue
   125:                 ;;
   126:             *.1|*.desktop|*.policy)
   127:                 check_tags
   128:                 check_lines
   129:                 check_fields
   130:                 check_trailing_spaces
   131:                 ;;
   132:             *.completion|*.sh)
   133:                 check_tags
   134:                 check_lines
   135:                 check_fields
   136:                 check_trailing_spaces
```

```
137:                 check_script_shellcheck
138:                 ;;
139:             *.py)
140:                 check_tags
141:                 check_lines
142:                 check_fields
143:                 check_trailing_spaces
144:                 check_script_pycodestyle
145:                 ;;
146:             *)
147:                 check_tags
148:                 check_lines
149:                 check_fields
150:                 check_trailing_spaces
151:                 check_script
152:                 check_usage
153:                 ;;
154:         esac
155:     done
156:     cd "$HOME" || exit $ERROR
157: }
158:
159:
160: check_tags() {
161:     if grep --quiet             \
162:             --word-regexp       \
163:             --regexp='#''XXX'   \
164:             --regexp='#''TODO'  \
165:             --regexp='#''FIXME' \
166:             "$script"; then
167:         info "
168: ${BOLD}In $script:${NORMAL}
169:     Gedit-tags gevonden.
170:
171: $(
172:     grep    --line-number           \
173:             --word-regexp           \
174:             --regexp='#''XXX'       \
175:             --regexp='#''TODO'      \
176:             --regexp='#''FIXME'     \
177:             "$script"               |
178:     nl      --number-width=4        \
179:             --number-separator=''   \
180:             --body-numbering=n
181: )
182: "
183:     fi
184: }
185:
186:
187: check_lines() {
188:     local -i line_num=0
189:     local -i max_lines=1000
190:     local -i max_line_length=79
191:     local -i max_line_length_found=0
192:
193:     case $script in
194:         *.sh)
195:             max_line_length=465
196:             ;;
197:         *.policy)
198:             max_line_length=106
199:             ;;
200:         *)
201:             max_line_length=79
202:             ;;
203:     esac
204:
```

```
  205:     max_line_length_found=$(wc --max-line-length < "$script")
  206:     if  [[ $max_line_length_found -gt $max_line_length ]]; then
  207:         info "
  208: ${BOLD}In $script:${NORMAL}
  209:     Een regel is langer dan $max_line_length ($max_line_length_found).
  210: "
  211:         if [[ $WARNING -gt $MAXRC ]]; then
  212:             MAXRC=$WARNING
  213:         fi
  214:     fi
  215:
  216:     line_num=$(wc --lines < "$script")
  217:     if [[ $line_num -gt $max_lines ]]; then
  218:         info "
  219: ${BOLD}In $script:${NORMAL}
  220:     Meer dan $max_lines regels ($line_num).
  221: "
  222:         if [[ $WARNING -gt $MAXRC ]]; then
  223:             MAXRC=$WARNING
  224:         fi
  225:     fi
  226:
  227:     # '#1 foo' = ok, '#1' = ok, '#1foo' = Â¬ok, hetzelfde voor #2, #3, 44n #4.
  228:     if [[ $script == kz-install*.sh || $script == kz-setup*.sh ]]; then
  229:         wrong_record=$(
  230:             grep --regexp='#[1-4]' "$script" --line-number  |
  231:             grep --regexp='#[1-4]$' --invert-match          |
  232:             grep --regexp='#[1-4] ' --invert-match          || true
  233:         )
  234:     else
  235:         wrong_record=''
  236:     fi
  237:     if [[ -n $wrong_record ]]; then
  238:         info "
  239: ${BOLD}In $script:${NORMAL}
  240:     Foutieve regel(s) gevonden.
  241: $(
  242:         printf  '%s'                    \
  243:                 "$wrong_record"         |
  244:         nl      --number-width=8        \
  245:                 --number-separator=''   \
  246:                 --body-numbering=n
  247:
  248:     )
  249: "
  250:         if [[ $ERROR -gt $MAXRC ]]; then
  251:             MAXRC=$ERROR
  252:         fi
  253:     fi
  254: }
  255:
  256:
  257: check_fields() {
  258:     if ! grep   --quiet                     \
  259:                 --regexp='RELEASE_YEAR='    \
  260:                 --regexp='release_year = '  \
  261:                 "$script"; then
  262:         info "
  263: ${BOLD}In $script:${NORMAL}
  264:     Vrijgavejaar (RELEASE_YEAR/release_year) ontbreekt.
  265: "
  266:         if [[ $ERROR -gt $MAXRC ]]; then
  267:             MAXRC=$ERROR
  268:         fi
  269:     fi
  270:     if ! grep   --quiet                         \
  271:                 --regexp='VERSION_NUMBER='      \
  272:                 --regexp='version_number = '    \
```

```
273:                 "$script"; then
274:         info "
275: ${BOLD}In $script:${NORMAL}
276:     Versienummer (VERSION_NUMBER/version_number) ontbreekt.
277: "
278:         if [[ $ERROR -gt $MAXRC ]]; then
279:             MAXRC=$ERROR
280:         fi
281:     fi
282:     if ! grep   --quiet                     \
283:                 --regexp='VERSION_DATE='    \
284:                 --regexp='version_date = '  \
285:                 "$script"; then
286:         info "
287: ${BOLD}In $script:${NORMAL}
288:     Versiedatum (VERSION_DATE/version_date) ontbreekt.
289: "
290:         if [[ $ERROR -gt $MAXRC ]]; then
291:             MAXRC=$ERROR
292:         fi
293:     fi
294:     if ! grep --quiet --word-regexp --regexp=' ''EOF' "$script"; then
295:         info "
296: ${BOLD}In $script:${NORMAL}
297:     EOF ontbreekt.
298: "
299:         if [[ $ERROR -gt $MAXRC ]]; then
300:             MAXRC=$ERROR
301:         fi
302:     fi
303: }
304:
305:
306: check_trailing_spaces() {
307:     if grep --quiet --regexp=' ''$' "$script"; then
308:         info "
309: ${BOLD}In $script:${NORMAL}
310:     Eindspaties gevonden.
311:
312: $(
313:     grep    --line-number --regexp=' ''$' "$script" |
314:     nl      --number-width=4  --number-separator='' --body-numbering=n
315:
316:     )
317: "
318:         if [[ $ERROR -gt $MAXRC ]]; then
319:             MAXRC=$ERROR
320:         fi
321:     fi
322: }
323:
324:
325: check_script() {
326:     if grep --quiet --line-regexp --regexp='#!''/bin/bash' "$script"; then
327:             check_script_shellcheck
328:     elif grep   --quiet                         \
329:                 --line-regexp                   \
330:                 --regexp='#!''/usr/bin/python3' \
331:                 "$script"; then
332:             check_script_pycodestyle
333:     else
334:         info "
335: ${BOLD}In $script:${NORMAL}
336:     Onbekend script.  Functie check_script() kan script niet controleren."
337:         if [[ $ERROR -gt $MAXRC ]]; then
338:             MAXRC=$ERROR
339:         fi
340:     fi
```

```
341: }
342:
343:
344: check_script_shellcheck() {
345:     local -i    check_rc=0
346:
347:     shellcheck --external-sources "$script" 2> >($LOGCMD) || check_rc=$?
348:     if [[ $check_rc -gt $MAXRC ]]; then
349:         MAXRC=$check_rc
350:     fi
351: }
352:
353:
354: check_script_pycodestyle() {
355:     local -i check_rc=0
356:
357:     pycodestyle "$script" || check_rc=$?
358:     if [[ $check_rc -gt $MAXRC ]]; then
359:         MAXRC=$check_rc
360:     fi
361: }
362:
363:
364: check_usage() {
365:     local -i check_rc=0
366:
367:     "$scriptsdir/$script" --version |& $LOGCMD || check_rc=$?
368:     if [[ $check_rc -gt $SUCCESS ]]; then
369:         info "
370: ${BOLD}In $script:${NORMAL}
371:     Uitvoeren '$script --version' gaat fout."
372:         if [[ $ERROR -gt $MAXRC ]]; then
373:             MAXRC=$ERROR
374:         fi
375:     fi
376:
377:     case $script in
378:         kz-backup)
379:             log "Check $script."
380:             "$scriptsdir/$script"   --target                \
381:                                     /tmp                    \
382:                                     /home/karel/kz-scripts  |& $LOGCMD  ||
383:             check_rc=$?
384:             if [[ $check_rc -gt $SUCCESS ]]; then
385:                 info "
386: ${BOLD}In $script:${NORMAL}
387:     Uitvoeren '$script --target /tmp /home/karel/kz-scripts' gaat fout."
388:                 if [[ $ERROR -gt $MAXRC ]]; then
389:                     MAXRC=$ERROR
390:                 fi
391:             fi
392:             ;;
393:         kz-install)
394:             log "Check $script."
395:             "$scriptsdir/$script" bitwarden |& $LOGCMD || check_rc=$?
396:             if [[ $check_rc -gt $SUCCESS ]]; then
397:                 info "
398: ${BOLD}In $script:${NORMAL}
399:     Uitvoeren '$script bitwarden' gaat fout."
400:                 if [[ $ERROR -gt $MAXRC ]]; then
401:                     MAXRC=$ERROR
402:                 fi
403:             fi
404:             "$scriptsdir/$script" --info |& $LOGCMD || check_rc=$?
405:             if [[ $check_rc -gt $SUCCESS ]]; then
406:                 info "
407: ${BOLD}In $script:${NORMAL}
408:     Uitvoeren '$script --info' gaat fout."
```

```
409:                 if [[ $ERROR -gt $MAXRC ]]; then
410:                     MAXRC=$ERROR
411:                 fi
412:             fi
413:             "$scriptsdir/$script" --remove |& $LOGCMD || check_rc=$?
414:             if [[ $check_rc -gt $SUCCESS ]]; then
415:                 info "
416: ${BOLD}In $script:${NORMAL}
417:     Uitvoeren '$script --remove' gaat fout."
418:                 if [[ $ERROR -gt $MAXRC ]]; then
419:                     MAXRC=$ERROR
420:                 fi
421:             fi
422:             ;;
423:         kz-restore)
424:             log "Check $script."
425:             printf                  '%s\n'          \
426:                                     '1'             |
427:             "$scriptsdir/$script"   --target        \
428:                                     /tmp            \
429:                                     --source /tmp   |& $LOGCMD || check_rc=$?
430:             if [[ $check_rc -gt $SUCCESS ]]; then
431:                 info "
432: ${BOLD}In $script:${NORMAL}
433:     Uitvoeren '$script --target /tmp --source /tmp' gaat fout."
434:                 if [[ $ERROR -gt $MAXRC ]]; then
435:                     MAXRC=$ERROR
436:                 fi
437:             fi
438:             sudo rm --force /tmp/backup_*.tar
439:             sudo rm --force --recursive /tmp/home
440:             ;;
441:         kz-setup)
442:             log "Check $script."
443:             "$scriptsdir/$script" bitwarden |& $LOGCMD || check_rc=$?
444:             if [[ $check_rc -gt $SUCCESS ]]; then
445:                 info "
446: ${BOLD}In $script:${NORMAL}
447:     Uitvoeren '$script bitwarden' gaat fout."
448:                 if [[ $ERROR -gt $MAXRC ]]; then
449:                     MAXRC=$ERROR
450:                 fi
451:             fi
452:             "$scriptsdir/$script" --info |& $LOGCMD || check_rc=$?
453:             if [[ $check_rc -gt $SUCCESS ]]; then
454:                 info "
455: ${BOLD}In $script:${NORMAL}
456:     Uitvoeren '$script --info' gaat fout."
457:                 if [[ $ERROR -gt $MAXRC ]]; then
458:                     MAXRC=$ERROR
459:                 fi
460:             fi
461:             "$scriptsdir/$script" --reset |& $LOGCMD || check_rc=$?
462:             if [[ $check_rc -gt $SUCCESS ]]; then
463:                 info "
464: ${BOLD}In $script:${NORMAL}
465:     Uitvoeren '$script --reset' gaat fout."
466:                 if [[ $ERROR -gt $MAXRC ]]; then
467:                     MAXRC=$ERROR
468:                 fi
469:             fi
470:             ;;
471:         *)
472:             return $SUCCESS
473:             ;;
474:     esac
475: }
476:
```

```
  477:
  478: term_script() {
  479:     exit $MAXRC
  480: }
  481:
  482:
  483: ##############################################################################
  484: # Main line
  485: ##############################################################################
  486:
  487: main() {
  488:     init_script "$@"
  489:     check_input "$@"
  490:     process_input
  491:     term_script
  492: }
  493:
  494:
  495: main "$@"
  496:
  497:
  498: # EOF
```