```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Checking scripts.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2015-2023.
##############################################################################

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

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


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

program_name='kz-check'
program_desc=$(gettext 'Checking scripts')
display_name=${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"


##############################################################################
# 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 {
    local script=''
    local scriptdir=''
    local scriptname=''
    local scripts_repo=$HOME/kz-scripts

    cd "$scripts_repo"
    while read -r script; do
        scriptdir=$(realpath "$(dirname "$script")")
        scriptname=$(basename "$script")
        check_trailing_spaces
        case $scriptname in
            *.policy)
                continue
                ;;
            *.1|*.desktop)
                check_record_length
                ;;
            kz_common.py)
                check_tags
                check_pycodestyle
                ;;
            kz_common.sh)
                check_tags
                check_record_length
                check_shellcheck
                ;;
            *.sh)
                check_tags
                check_record_types
                check_shellcheck
                ;;
            *)
                check_tags
                check_record_length
                check_code
                ;;
        esac
    done < <(
            find .                          \
            -type f                         \
            -not -path './.git*'            \
            -not -path '*/__pycache__*' \
            -not -name kz.mo                \
            -not -name kz.po                \
            -not -name kz.pot               \
            -not -name LICENSE              \
            -not -name README.md            \
            -print                          |
            sort
            )
    cd "$HOME"
```

```
}


function check_code {
    local bash_script='#!''/bin/bash'
    local python_script='#!''/usr/bin/python3'
    local tab_completion_script='# ''shellcheck shell=bash'

    if grep --quiet --line-regexp --regexp="$bash_script" "$script"; then
        check_shellcheck
    elif grep --quiet --line-regexp --regexp="$tab_completion_script" "$script"
        then
        check_shellcheck
    elif grep --quiet --line-regexp --regexp="$python_script" "$script"; then
        check_pycodestyle
    else
        error "
In $scriptdir / $scriptname:
$(gettext "    Unknown script. Function check_code cannot check script code.")"
        maxrc=$error
    fi
}


function check_pycodestyle {
    local -i rc=0

    pycodestyle "$script" || rc=$?
    if [[ $rc -ne $ok ]]; then
        maxrc=$error
    fi
}


function check_record_length {
    local -i max_line_length=79
    local -i max_line_length_found=0
    local python_script='#!''/usr/bin/python3'

    # Python is checked by pycodestyle.
    if grep --quiet --line-regexp --regexp="$python_script" "$script"; then
        return $ok
    fi

    max_line_length_found=$(wc --max-line-length < "$script")
    if  [[ $max_line_length_found -gt $max_line_length ]]; then
        error "
In $scriptdir / $scriptname:
$(eval_gettext "    A line is longer than \$max_line_length \
(\$max_line_length_found).")
"
        maxrc=$error
    fi
}


function check_record_types {
    local wrong_record=''

    # Ok:
    # # APP             #-APP           # APP             #-APP
    # # HOST            # HOST          # USER            # USER
    # # <Description>   # <Description> # <Description>   # <Description>
    # Wrong:
```

```
    # The rest :-)
    wrong_record=$(
        grep --regexp='^# APP' --after-context=2 --line-number "$script" |
        grep --regexp='# HOST ' --invert-match                            |
        grep --regexp='# USER ' --invert-match                            |
        grep --regexp='# '      --invert-match                            |
        grep --regexp='--'      --invert-match                           || true
        )
    if [[ -n $wrong_record ]]; then
        error "
In $scriptdir / $scriptname:
    $(gettext 'Faulty line(s) found.')
$(
    printf '%s' "$wrong_record" |
                nl --number-width=8 --number-separator='' --body-numbering=n
)
"
        maxrc=$error
    fi
}


function check_shellcheck {
    local -i rc=0

    # Shellcheck doesn't work well with absolute filenames.
    cd "$scriptdir"
    shellcheck  --enable=add-default-case           \
                --enable=avoid-nullary-conditions   \
                --enable=check-set-e-suppressed     \
                --enable=deprecate-which            \
                --enable=require-double-brackets    \
                --external-sources                  \
                "$scriptname"                       || rc=$?
    if [[ $rc -ne $ok ]]; then
        maxrc=$error
    fi
    cd "$scripts_repo"
}


function check_tags {
    local bug='B''U''G'
    local fixme='F''I''X''M''E'
    local todo='T''O''D''O'

    if grep --quiet             \
            --word-regexp       \
            --regexp="$bug"     \
            --regexp="$fixme"   \
            --regexp="$todo"    \
            "$script"; then
        warning "
In $scriptdir / $scriptname:
$(gettext '    Flagged annotation found.')

$(
    grep    --line-number           \
            --word-regexp           \
            --regexp="$bug"         \
            --regexp="$fixme"       \
            --regexp="$todo"        \
            "$script"               |
    nl      --number-width=4        \
```

```
            --number-separator=''   \
            --body-numbering=n
)
"
    fi
}


function check_trailing_spaces {
    if grep --quiet --regexp=' ''$' "$script"; then
        error "
In $scriptdir / $scriptname:
$(gettext '    End spaces found.')
$(
    grep    --line-number --regexp=' ''$' "$script" |
    nl      --number-width=4  --number-separator='' --body-numbering=n
)
"
        maxrc=$error
    fi
}


function term_script {
    exit $maxrc
}


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

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

main "$@"
```