```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Installation menu.
#
# Written in 2015 by Karel Zimmer <info@karelzimmer.nl>, Creative Commons
# Public Domain Dedication <https://creativecommons.org/publicdomain/zero/1.0>.
##############################################################################

set -o errexit
program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)
source "$program_path"/kz_common.sh


##############################################################################
# Constants
##############################################################################

declare program_name='kz-menu'
declare program_desc
program_desc=$(gettext 'Installation menu')
declare display_name=${program_name/kz-/kz }

declare usage
usage="$(eval_gettext "Usage: \$display_name [-g|--gui]")
                $options_usage"
declare help
help="$(eval_gettext "Usage: \$display_name [OPTION...]")

$program_desc.

$(gettext 'Options:')
  -g, --gui      $(gettext 'starts in graphics mode')
$options_help"
declare options_short+='g'
declare options_long+=',gui'

declare menu1_title
menu1_title=$(gettext 'Installation menu')
declare menu1_head="                  $menu1_title"
declare -a menu1_actions=(
            [1]="$(gettext 'Preparing installation')"
                "$(gettext 'Perform installation')"
                "$(gettext 'Finish installation')"
                "$(gettext 'Provision user')"
                "$(gettext 'Commands menu')"
                "$(gettext 'Stop')"
            )
declare -a menu1_commands=(
            [1]="$(gettext 'Checklist chapter 1')"
                "$(gettext 'Checklist chapter 2')"
                "$(gettext 'Checklist chapter 3')"
                "$(gettext 'Checklist chapter 4')"
                "$(gettext 'Used commands')"
                "$(gettext 'Exit menu')"
            )
declare menu1_lines=${#menu1_actions[@]}

declare menu2_title
menu2_title=$(gettext 'Commands menu')
declare menu2_head="             $menu2_title"
declare -a menu2_actions=(
            [1]="$(gettext 'Show WiFi data')"
                "$(gettext 'Create backup')"
```

```
                "$(gettext 'Add users')"
                "$(gettext 'Install apps')"
                "$(gettext 'Restore backup')"
                "$(gettext 'Set user photo')"
                "$(gettext 'Set up apps')"
                "$(gettext 'Back')"
            )
declare -a menu2_commands=(
            [1]='kz wifi'
                'kz backup'
                "$(gettext 'Manually')"
                'kz install'
                'kz restore'
                "$(gettext 'Manually')"
                'kz setup'
                "$(gettext 'Previous menu')"
            )
declare menu2_lines=${#menu2_actions[@]}


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

declare execute_command=false
declare -a rows=()
declare option_gui=false


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

function check_input {
    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 $ok ]]; then
        info "$usage_line"
        exit $error
    fi
    eval set -- "$parsed"
    kz_common.process_options "$@"

    while true; do
        case $1 in
            -g|--gui)
                option_gui=true
                kz_common.reset_terminal_attributes
                shift
                ;;
            --)
                shift
                break
                ;;
            *)
                shift
```

```
                ;;
        esac
    done

    if [[ "$*" ]]; then
        printf  "$display_name: $*: %s\n$usage_line\n" \
                "$(gettext 'arguments are not allowed')"
        exit $error
    fi
}


function process_input {
    if $option_gui; then
        process_menu1_gui
    else
        process_menu1_tui
    fi
}


function process_menu1_gui {
    local -i zenity_rc=0

    while true; do
        title=$menu1_title
        REPLY=$(
            zenity  --list                                              \
                    --radiolist                                         \
                    --width         480                                 \
                    --height        240                                 \
                    --title         "$title"                            \
                    --text          "$(gettext 'Select a choice:')"     \
                    --ok-label      "$(gettext 'Continue')"             \
                    --cancel-label  "$(gettext 'Exit')"                 \
                    --column        "$(gettext 'Choice')"               \
                    --column        "$(gettext 'No.')"                  \
                    --column        "$(gettext 'Action')"               \
                    --column        "$(gettext 'Explanation')"          \
                    TRUE  1 "${menu1_actions[1]}" "${menu1_commands[1]}" \
                    FALSE 2 "${menu1_actions[2]}" "${menu1_commands[2]}" \
                    FALSE 3 "${menu1_actions[3]}" "${menu1_commands[3]}" \
                    FALSE 4 "${menu1_actions[4]}" "${menu1_commands[4]}" \
                    FALSE 5 "${menu1_actions[5]}" "${menu1_commands[5]}" \
                    2> >($logcmd)
            ) || zenity_rc=$?
        if [[ $zenity_rc -ne 0 ]]; then
            exit $ok
        fi
        process_menu1_choice
    done
}


function process_menu1_tui {
    local action=''

    while true; do

        # Show menu.
        clear -x
        info "$menu1_head
"
        for action in "${!menu1_actions[@]}"; do
```

```
            printf  '%2s. %-24.24s - %s\n'      \
                    "$action"                   \
                    "${menu1_actions[$action]}" \
                    "${menu1_commands[$action]}"
        done
        info ''

        while true; do
            read -rp "$(eval_gettext "Run number [1-\$menu1_lines]: ")"
            case $REPLY in
                *[^[:digit:]]*)
                    printf '%s' "${rewrite_line}"
                    continue
                    ;;
                *)
                    if [[ $REPLY -ge 1 && $REPLY -le $menu1_lines ]]; then
                        break
                    else
                        printf '%s' "${rewrite_line}"
                        continue
                    fi
                    ;;
            esac
        done
        process_menu1_choice
        execute_command=false
    done
}


function process_menu1_choice {
    case $REPLY in
        1)
            process_menu1_choice1
            ;;
        2)
            process_menu1_choice2
            ;;
        3)
            process_menu1_choice3
            ;;
        4)
            process_menu1_choice4
            ;;
        5)
            process_menu2
            ;;
        6)
            exit $ok
            ;;
        *)
             error "$(eval_gettext "Cannot handle this choice (\$REPLY).")"
            exit $error
            ;;
    esac
}


function process_menu1_choice1 {
    title="${menu1_actions[1]}"
    rows=(
        [1]="${menu2_actions[1]}#${menu2_commands[1]}"
            "${menu2_actions[2]}#${menu2_commands[2]}"
        )
```

```
    process_commands
}


function process_menu1_choice2 {
    title="${menu1_actions[2]}"
    text="
$(gettext "Follow the steps as described in Installation Checklist, Chapter 2.
Checklist installation can be found on the site https://karelzimmer.nl,
under Linux.

Roughly boils down to:
1. Download an image file (.iso).
2. Create a bootable Live USB stick or DVD.
3. Boot the computer from this USB stick or DVD.
4. Install Linux.")"
    info "$text"
    kz_common.wait_for_enter
}


function process_menu1_choice3 {
    title="${menu1_actions[3]}"
    rows=(
        [1]="${menu2_actions[3]}#${menu2_commands[3]}"
            "${menu2_actions[4]}#${menu2_commands[4]}"
        )
    process_commands
}


function process_menu1_choice4 {
    title="${menu1_actions[4]}"
    rows=(
        [1]="${menu2_actions[5]}#${menu2_commands[5]}"
            "${menu2_actions[6]}#${menu2_commands[6]}"
            "${menu2_actions[7]}#${menu2_commands[7]}"
        )
    process_commands
}


function process_menu2 {
    if $option_gui; then
        process_menu2_gui
    else
        process_menu2_tui
    fi
}


function process_menu2_gui {
    local -i zenity_rc=0

    while true; do
        title=$menu2_title
        REPLY=$(
            zenity  --list                                              \
                    --radiolist                                         \
                    --width         480                                 \
                    --height        290                                 \
                    --title         "$title"                            \
                    --text          "$(gettext 'Select a choice:')"     \
                    --ok-label      "$(gettext 'Continue')"             \
```

```
                    --cancel-label  "$(gettext 'Back')"                    \
                    --column        "$(gettext 'Choice')"                  \
                    --column        "$(gettext 'No.')"                     \
                    --column        "$(gettext 'Action')"                  \
                    --column        "$(gettext 'Command')"                 \
                    TRUE   1 "${menu2_actions[1]}"  "${menu2_commands[1]}"  \
                    FALSE  2 "${menu2_actions[2]}"  "${menu2_commands[2]}"  \
                    FALSE  3 "${menu2_actions[3]}"  "${menu2_commands[3]}"  \
                    FALSE  4 "${menu2_actions[4]}"  "${menu2_commands[4]}"  \
                    FALSE  5 "${menu2_actions[5]}"  "${menu2_commands[5]}"  \
                    FALSE  6 "${menu2_actions[6]}"  "${menu2_commands[6]}"  \
                    FALSE  7 "${menu2_actions[7]}"  "${menu2_commands[7]}"  \
                    2> >($logcmd)
            ) || zenity_rc=$?
        if [[ $zenity_rc -ne 0 ]]; then
            break
        fi
        process_menu2_choice
    done
}


function process_menu2_tui {
    local action=''
    local exit_menu2=false

    while true; do

        # Show menu.
        clear -x
        info "$menu2_head
"
        for action in "${!menu2_actions[@]}"; do
            printf  '%2s. %-24.24s - %s\n'      \
                    "$action"                   \
                    "${menu2_actions[$action]}" \
                    "${menu2_commands[$action]}"
        done
        info ''

        while true; do
            read -rp "$(eval_gettext "Run number [1-\$menu1_lines]: ")"
            case $REPLY in
                *[^[:digit:]]*)
                    printf '%s' "${rewrite_line}"
                    continue
                    ;;
                *)
                    if [[ $REPLY -ge 1 && $REPLY -le $menu2_lines ]]; then
                        break
                    else
                        printf '%s' "${rewrite_line}"
                        continue
                    fi
                    ;;
            esac
        done
        process_menu2_choice
        if $exit_menu2; then
            break
        fi
        execute_command=false
    done
}
```

```
function process_menu2_choice {
    if [[ $REPLY -ge 1 && $REPLY -le $((menu2_lines -1)) ]]; then
        title="${menu2_actions[$REPLY]}"
        rows=(
            [1]="${menu2_actions[$REPLY]}#${menu2_commands[$REPLY]}"
            )
        process_commands
    elif [[ $REPLY -eq $menu2_lines ]]; then
        exit_menu2=true
    else
         error "$(eval_gettext "Cannot handle this choice (\$REPLY).")"
        exit $error
    fi
}


function process_commands {
    local action=''
    local command=''
    local -i row=0

    if ! $option_gui; then
        clear -x
    fi
    if [[ ${#rows[@]} -eq 1 ]]; then
        execute_command=true
    else
        show_commands_before_execution
    fi
    if $execute_command; then
        for row in "${!rows[@]}"; do
            action=$(
                printf '%s' "${rows[$row]}" | cut --delimiter='#' --fields=1
                )
            command=$(
                printf '%s' "${rows[$row]}" | cut --delimiter='#' --fields=2
                )
            title=$action
            execute_command
        done
    fi
    if ! $option_gui; then
        clear -x
    fi
    if $execute_command && [[ ${#rows[@]} -gt 1 ]]; then
        show_commands_after_execution
    fi
}


function show_commands_before_execution {
    local action=''
    local command=''
    local -i row=0
    local prompt=''

    text="$(gettext 'The following will be executed:')
"
    for row in "${!rows[@]}"; do
        action=$(
            printf '%s' "${rows[$row]}" | cut --delimiter='#' --fields=1
            )
```

```
        command=$(
            printf '%s' "${rows[$row]}" | cut --delimiter='#' --fields=2
            )
        text+="
$row. $action ($command)"
    done
    text+='
'
    prompt="$(gettext 'Proceed?')"
    if $option_gui; then
        text+="
$prompt"
        if zenity   --question                          \
                    --no-markup                         \
                    --width         600                 \
                    --height        100                 \
                    --title         "$title"            \
                    --text          "$text"             \
                    --ok-label      "$(gettext 'Yes')"  \
                    --cancel-label  "$(gettext 'No')"   2> >($logcmd); then
            execute_command=true
        else
            execute_command=false
        fi
    else
        info "$text"
        while true; do
            read -rp "$(eval_gettext "\$prompt [Y/n]: ")"
            case $REPLY in
                y*|Y*|j*|J*|'')
                    execute_command=true
                    break
                    ;;
                n*|N*)
                    execute_command=false
                    break
                    ;;
                *)
                    printf '%s' "${rewrite_line}"
                    continue
                    ;;
            esac
        done
    fi
}


function execute_command {
    local -i cmd_rc=0
    local row=''
    local gui_option=''

    if ! $option_gui; then
        clear -x
    fi

    # Handle manual actions.
    if [[ $command = $(gettext 'Manually') ]]; then
        manual_actions
        return $ok
    fi

    # Process (automatic) commands.
    row=${command/kz /kz-}
```

```
    cmd=$(
        printf '%s' "$row" | cut --delimiter=' ' --fields=1
        )
    arg=$(
        printf '%s' "$row" | cut --delimiter=' ' --fields=2- --only-delimited
        )
    if $option_gui; then
        gui_option='--gui'
    fi
    "$program_path/$cmd" "$arg" $gui_option -- || cmd_rc=$?
    log "$command: rc=$cmd_rc"
    kz_common.wait_for_enter
}


function manual_actions {
    if [[ $title = $(gettext 'Add users') ]]; then
        text="$(gettext 'Add users'):

$(gettext "1. Add any additional Standard or Admin users by pressing the \
Super key[1],
   type 'user' and click behind Settings on Users.
2. Click Unlock and enter password.
3. Click Add User.
4. Complete the screen and click Add.
5. Close the Users screen.

[1] The Super key is the Windows key, Command key (Apple), or Magnifier key \
(Chromebook).")"
        info "$text"
        kz_common.wait_for_enter
    elif [[ $title = $(gettext 'Set user photo') ]]; then
        if [[ -f $HOME/$(gettext 'Settings')/$(gettext 'Userphoto') ]]; then
            text="$(gettext 'Set user photo'):

$(gettext "1. Change the user photo by pressing the Super key[1], type 'user' \
and click
   behind Settings on Users.
2. Click Unlock.
3. Click the image for the user.
4. Click Select a file.
5. Select Userphoto in Personal folder / Settings.
6. Close the Users screen.

[1] The Super key is the Windows key, Command key (Apple), or Magnifier key \
(Chromebook).")"
            info "$text"
        else
            text="$(gettext 'Set user photo'):
$(gettext 'No user photo was found when backing up.')"
            info "$text"
        fi
        kz_common.wait_for_enter
    fi
}


function show_commands_after_execution {
    local action=''
    local command=''
    local -i row=0

    text="$(gettext 'The following has been executed:')
"
```

```
    for row in "${!rows[@]}"; do
        action=$(
            printf '%s' "${rows[$row]}" | cut --delimiter='#' --fields=1
            )
        command=$(
            printf '%s' "${rows[$row]}" | cut --delimiter='#' --fields=2
            )
        text+="
$row. $action ($command)"
    done
    info "$text"
    kz_common.wait_for_enter
}


function term_script {
    exit $ok
}


################################################################################
# Script
################################################################################

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

main "$@"
```