

```

#!/bin/bash
# shellcheck source=kz_common.sh
#####
# Install apps.
#
# Written in 2009 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-install'
declare program_desc
program_desc=$(gettext 'Install apps')
declare display_name=${program_name/kz-/kz }

declare usage
usage=$(eval_gettext "Usage: \${display_name} [-a|--apps] [-c|--cat] \
[-f|--files] [-g|--gui]
[-r|--remove] [--server] [-s|--simulate]
\${options_usage}
[APP...] [FILE...]" )

declare help
help="$(eval_gettext "Usage: \${display_name} [OPTION...] [APP...] [FILE...]" )

$program_desc.

$(gettext 'Options:')
-a, --apps      $(gettext 'show list of available apps')
-c, --cat       $(gettext 'show contents of files')
-f, --files     $(gettext 'show list of files')
-g, --gui       $(gettext 'starts in graphics mode')
-r, --remove    $(gettext 'delete apps')
--server        $(gettext 'install apps for a server')
-s, --simulate  $(gettext 'show commands')
$options_help

$(gettext 'Arguments:
APP            install APPs
FILE           process FILEs')"
declare options_short+='acfgsr'
declare options_long+=", apps, cat, files, gui, remove, server, simulate"

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

declare -a app_arguments=()
declare apps_file=''
declare argument_app=false
declare argument_file=false
declare cmds_file=''
declare commands_found=false
declare distro=''
declare edition=desktop
declare execute_commands=true

```

```

declare -a file_arguments=()
declare -a files_to_process=()
declare -i maxrc=0
declare option_apps=false
declare option_cat=false
declare option_files=false
declare option_gui=false
declare option_remove=false
declare option_simulate=false
declare sims_file=''

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

function check_input {
    local -i app_arg_num=0
    local -i file_arg_num=0
    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
            -a|--apps)
                option_apps=true
                shift
                ;;
            -c|--cat)
                option_cat=true
                shift
                ;;
            -f|--files)
                option_files=true
                shift
                ;;
            -g|--gui)
                option_gui=true
                kz_common.reset_terminal_attributes
                shift
                ;;
            -r|--remove)
                option_remove=true
                shift
                ;;
            --server)
                edition=server
                shift
                ;;
            -s|--simulate)

```

```

        option_simulate=true
        execute_commands=false
        shift
        ;;
    --)
        shift
        break
        ;;
    *)
        shift
        ;;
    esac
done

# Check arguments.
while [[ "$*" ]]; do
    if [[ "$(basename "$1")" == $program_name-*.sh ]]; then
        argument_file=true
        file_arguments[file_arg_num]=$1
        ((++file_arg_num))
        shift
    else
        argument_app=true
        app_arguments[app_arg_num]=$1
        ((++app_arg_num))
        shift
    fi
done
}

function process_input {
    local dfltfile_1=''
    local dfltfile_2=''
    local file_notfound=false
    local -i file_num=0
    local file=''

    # Create temporary files.
    apps_file=$(mktemp -t "$program_name-XXXXXXXXXX.apps")
    cmds_file=$(mktemp -t "$program_name-XXXXXXXXXX.cmds")
    sims_file=$(mktemp -t "$program_name-XXXXXXXXXX.sims")

    # Process arguments.
    distro=$(lsb_release --id --short | tr '[:upper:]' '[:lower:]')
    dfltfile_1=$program_path/$program_name-$distro-$edition.sh
    dfltfile_2=$program_path/$program_name-$distro-$edition-$HOSTNAME.sh
    if $argument_app; then
        process_argument_app
    fi
    if $argument_file; then
        process_argument_file
    fi
    # Somewhere before this it didn't go very well.
    if $file_notfound; then
        exit $error
    fi
    # Do not process standard files for certain arguments and options.
    if ! ($argument_app || $argument_file || $option_files); then
        process_dfltfiles
    fi

    # Process other options; depending on files and arguments.
    if $option_apps; then

```

```

        process_option_apps
        exit $ok
    elif $option_cat; then
        process_option_cat
        exit $ok
    elif $option_files; then
        process_option_files
        exit $ok
    fi

# Opdrachten uitvoeren of simuleren.
if $execute_commands; then
    kz_common.check_user_root
    kz_common.check_on_ac_power
fi

for file in "${files_to_process[@]}"; do
    cat "$file" >> "$cmds_file"
done

if $execute_commands && $option_gui; then
    title=$(gettext 'Install apps')
    text=$(gettext 'Prepare to install')
    # With '|& zenity --progress' global variables from called functions
    # are not passed, hence the process substitution with '> >(zenity...)'.
    process_cmds_file > >(
        zenity --progress          \
              --pulsate            \
              --auto-close         \
              --no-cancel          \
              --width 700          \
              --height 50          \
              --title "$title"     \
              --text "$text"       2> >($logcmd)
    )
else
    process_cmds_file
fi
}

function process_argument_app {
    local app=''
    local file=''
    local -a files=()

    # Search for available APPs in all installation files.
    for file in "$program_path/$program_name-".*.sh; do
        files+=("$file")
    done
    for app in "${app_arguments[@]}"; do
        file=$(
            grep --files-with-matches \
                --word-regexp \
                --regexp="#1 $app" \
                "${files[@]}" \
                2> >($logcmd) \
            head --lines=1 || printf '\n'
        )
        if [[ $file ]]; then
            process_argument_app_file
        else
            printf "$display_name: %s: %s\n" \
                "$app" "$(gettext 'app not found')"
        fi
    done
}

```

```

        file_notfound=true
    fi
done
if $file_notfound; then
    printf "%s\n" \
        "$$(gettext "Type '\$display_name --apps' for available apps.*)"
else
    files_to_process[file_num]=$apps_file
    ((++file_num))
fi
}

```

```

function process_argument_app_file {
    local app_found=false
    local record=''

    while read -r record; do
        case $record in
            '#1 '* )
                if [[ $record == '#1 '$app* ]]; then
                    # Found app name.
                    # Write records to temporary file until next app name or
                    # until eof.
                    printf '%s\n' "$record" >> "$apps_file"
                    app_found=true
                elif $app_found; then
                    # Next app.
                    break
                fi
                ;;
            '#1-' '* )
                if $app_found; then
                    # Next (hidden) app.
                    break
                fi
                ;;
            *)
                # Other records.
                if $app_found; then
                    printf '%s\n' "$record" >> "$apps_file"
                fi
                ;;
        esac
    done < "$file"
}

```

```

function process_argument_file {
    local file=''

    for file in "${file_arguments[@]}"; do
        if [[ -f $file ]]; then
            files_to_process[file_num]=$file
            ((++file_num))
        else
            printf "$display_name: %s: %s\n" \
                "$file" "$$(gettext 'file not found')"
            file_notfound=true
        fi
    done
    if $file_notfound; then
        printf "%s\n" "$$(gettext "Type '\$display_name --files' for available \
files.*)"
    fi
}

```

```

    fi
}

function process_dfltfiles {
    if [[ -f $dfltfile_1 ]]; then
        files_to_process[file_num]=$dfltfile_1
        ((++file_num))
    else
        warning "$(gettext 'No installation files found.')"
        exit $error
    fi
    if [[ -f $dfltfile_2 ]]; then
        files_to_process[file_num]=$dfltfile_2
        ((++file_num))
    fi
}

function process_option_apps {
    local -a files=()

    # Search for available APPs in all installation files.
    for file in "$program_path/$program_name- "*.sh; do
        files+=("$file")
    done

    text="$(gettext 'The following APPs are available:

    APP')
$(
    if ! grep --no-messages \
        --regexp='^#1 ' \
        "${files[@]}" \
        cut --delimiter=' ' \
        --fields=2- \
        sort --unique \
        nl --number-width=2 \
        --number-format=rn \
        --number-separator='] ' \
        --body-numbering=a \
        sed --expression='s/^[/'; then
        printf '%s\n' "$(gettext ' 0 No apps found.')"

$usage_line"
    else
        printf '%s\n' "
$(gettext 'To install the APPs run:') ${blue}$display_name APP...${normal}
$(gettext 'To simulate installing APPs execute:') ${blue}$display_name \
--simulate APP...${normal}
$(gettext 'To view the contents of the APPs installation file execute:') \
${blue}$display_name --cat APP...${normal}"
    fi
)"
    info "$text"
}

function process_option_cat {
    local first_file=true

    for file in "${files_to_process[@]"; do
        {
            if $first_file; then

```

```

        first_file=false
    else
        printf '\n'
    fi
    printf "${blue}%s\n${normal}" "# $file"
    cat "$file"
} >> "$cmds_file"
done
less "$less_options" "$cmds_file"
}

```

```

function process_option_files {
    text="$(gettext 'The following FILES are available:

```

```

    FILE')
$(
    if ! find "$program_path/$program_name-*" \
        2> >($logcmd) |
        nl --number-width=2 \
        --number-format=rn \
        --number-separator=] ' \
        --body-numbering=a |
        sed --expression='s/^/[/'; then
        printf '%s\n' "$(eval_gettext ' 0 No files present.')"

```

```

$usage_line"
    else
        printf '%s\n' "
$(gettext 'To process the FILES execute'): ${blue}$display_name \
FILE...${normal}
$(gettext 'To view the contents of FILES execute'): ${blue}$display_name \
--cat FILE...${normal}
$(gettext 'To simulate the processing of FILES execute'): \
${blue}$display_name --simulate FILE...${normal}"
    fi
)"
    info "$text"
}

```

```

function process_cmds_file {
    local app_description=''
    local first_description_line=true
    local operation
    operation=$(gettext 'install')
    local record=''
    local recordtype=''
    local write_app_description_line=true
    local -i app_seq_num=0
    local -i app_tot_num=0
    local -i cmd_seq_num=0

    app_tot_num=$(
        grep --word-regexp --regexp='^#1' --count "$cmds_file" || true
    )
    if [[ $app_tot_num -gt 99 ]]; then
        app_tot_num=99
    fi

    if $option_remove; then
        operation=$(gettext 'remove')
    fi
    if $option_simulate; then

```

```

        operation+=$(gettext ' simulate')
    fi

while read -r record; do
    recordtype=${record:0:2}
    case $recordtype in
        '')
            # Empty line or empty app name.
            continue
            ;;
        '#1')
            # App name and description.
            process_app_description_record "$record"
            ;;
        '#2')
            # Remove command.
            if $option_remove; then
                process_command_record "${record:3}"
            fi
            ;;
        '#')
            # Comment line.
            continue
            ;;
        *)
            # Install command.
            if ! $option_remove; then
                process_command_record "$record"
            fi
            ;;
    esac
done < "$cmds_file"
}

function process_app_description_record {
    local record=${1:-unknown}

    app_description="${record:3} $operation"
    write_app_description_line=true
    ((++app_seq_num))
    if [[ $app_seq_num -gt 99 ]]; then
        app_seq_num=99
    fi

    cmd_seq_num=0
}

function process_command_record {
    local cmd=${1:-unknown}
    local app_description_line=''
    local -i cmd_rc=0

    commands_found=true
    ((++cmd_seq_num))
    if [[ $cmd_seq_num -gt 99 ]]; then
        cmd_seq_num=99
    fi

    if $execute_commands; then
        if $option_gui; then
            text="#"[$app_seq_num/$app_tot_num] $app_description\n\n$cmd"
            printf '%s\n' "$text"
        fi
    fi
}

```



```

fi
printf -v app_description_line \
    "[%2d/%-2d] [%2d] %s" \
    "$app_seq_num" \
    "$app_tot_num" \
    "$cmd_seq_num" \
    "$app_description"
execute_command "$cmd"
else
# option_simulate
if $write_app_description_line; then
    if $first_description_line; then
        first_description_line=false
    else
        printf '\n' >> "$sims_file"
    fi
    printf "[%2d/%-2d] %s\n" \
        "$app_seq_num" \
        "$app_tot_num" \
        "$app_description" >> "$sims_file"
    write_app_description_line=false
fi
printf '%7s [%2d] %s\n' \
    " " \
    "$cmd_seq_num" \
    "${blue}$cmd${normal}" >> "$sims_file"
fi
}

```

```

function execute_command {
    local cmd=${1:-unknown}
    local -i cmd_rc=0
    local status=''

    if ! $option_gui; then
        if [[ $cmd_seq_num -eq 1 ]]; then
            printf "%b\n" "$app_description_line"
        else
            printf "      %b\n" "${app_description_line:7}"
        fi
    fi

    log "$app_description_line"
    log "${blue}$cmd${normal}"
    kz_common.check_dpkgd_snapd
    eval "$cmd" |& $logcmd || cmd_rc=$?
    if [[ $cmd_rc -gt $maxrc ]]; then
        maxrc=$cmd_rc
    fi
    if [[ $cmd_rc -eq $ok ]]; then
        status='ok'
    else
        status='error'
    fi
    log "$status rc=$cmd_rc, maxrc=$maxrc"
}

```

```

function term_script {
    local action
    action=$(gettext 'install')

    if ! $commands_found; then

```

```
        info "$(gettext 'No commands to execute.' )"
    elif $execute_commands; then
        if [[ $maxrc -ne $ok ]]; then
            error "$(eval_gettext "One or more commands went wrong.
The maximum exit value is \ $maxrc.
Check the log in the Terminal with the command:") \
${blue}$logcmd_check${normal}"
            exit $error
        else
            if $option_remove; then
                # shellcheck disable=SC2034
                action=$(gettext 'remove')
            fi
            info "
$(eval_gettext "Apps \ $action completed.")"
            fi
        else
            less "$less_options" "$sims_file"
        fi
        exit $ok
    }
}
```

```
#####
# Script
#####
```

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

```
main "$@"
```