

```

#!/bin/bash
# shellcheck source=kz_common.sh
#####
# Set up apps.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2013-2023.
#####
#####
# Import
#####

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

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

program_name='kz-setup'
program_desc=$(gettext 'Set up apps')
display_name=${program_name/kz-/kz }

usage=$(eval_gettext "Usage: \${display_name} [-a|--apps] [-c|--cat] \
[-f|--file FILE] [-g|--gui]
                    [-l|--list] [-r|--reset] [--server] [-s|--simulate]
                    \${options_usage}
                    [--] [APP...]"
help=$(eval_gettext "Usage: \${display_name} [OPTION...] [APP...]"

$program_desc.

$(gettext 'Options:')
  -a, --apps      $(gettext 'show list of available apps')
  -c, --cat       $(gettext 'show contents of file')
$(gettext '  -f, --file FILE
                    process FILE')
  -g, --gui       $(gettext 'starts in graphics mode')
  -l, --list      $(gettext 'show list of standard files')
  -r, --reset     $(gettext 'set apps on default values')
  --server        $(gettext 'setup apps for a server')
  -s, --simulate  $(gettext 'show commands')
$options_help
  --              $(gettext "signals the end of options and disables further \
option processing")

$(gettext 'Arguments:
  APP              setup APPs')"
options_short+='acf:lgrs'
options_long+=",apps,cat,file:,list,gui,reset,server,simulate"

declare -a app_arguments=()
argument_app=false
cmdsfile=''
commands_found=false
distro=''
edition='desktop'
execute_commands=true
file_argument=''
mode='setup'
option_apps=false
option_cat=false

```

```
option_file=false
option_list=false
option_gui=false
option_reset=false
option_simulate=false
```

```
cmdsfile=$(mktemp -t "$program_name-$mode-cmds-XXXXXXXXXX")
simsfile=$(mktemp -t "$program_name-$mode-sims-XXXXXXXXXX")
```

```
#####
# Functions
#####
```

```
function check_input {
    local -i app_arg_num=0
    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
            -a|--apps)
                option_apps=true
                shift
                ;;
            -c|--cat)
                option_cat=true
                shift
                ;;
            -f|--file)
                if $option_file; then
                    printf "$display_name: $1 $2: %s\n$usage_line\n" \
                        "$(gettext 'too many options')"
                    exit $error
                fi
                option_file=true
                file_argument=$2
                shift 2
                ;;
            -l|--list)
                option_list=true
                shift
                ;;
            -g|--gui)
                option_gui=true
                reset_terminal_attributes
                shift
                ;;
            -r|--reset)
                option_reset=true
        esac
    done
}
```

```

        mode='reset'
        shift
        ;;
    --server)
        edition=server
        shift
        ;;
    -s|--simulate)
        option_simulate=true
        execute_commands=false
        shift
        ;;
    --)
        shift
        break
        ;;
    *)
        shift
        ;;
esac
done

# Check arguments.
while [[ -n "$*" ]]; do
    argument_app=true
    app_arguments[app_arg_num]=$1
    ((++app_arg_num))
    shift
done
}

function process_input {
    local inputfile=''
    local file_not_found=false
    local file=''

    if $option_file; then
        if ! [[ "$(basename "$file_argument")" == "$program_name-*.sh" ]]; then
            text=$(eval_gettext "File name must start with '\$program_name-'.")
            warning "$text"
            exit $error
        fi
        inputfile=$file_argument
    else
        distro=$(lsb_release --id --short | tr '[:upper:]' '[:lower:]')
        inputfile=$program_path/$program_name-$mode-$distro-$edition.sh
    fi
    if ! [[ -f $inputfile ]]; then
        text=$(eval_gettext "No install file found, looked for \$inputfile.")
        warning "$text"
        exit $error
    fi

    if $argument_app; then
        process_argument_app
    else
        process_inputfile
    fi

    if $option_apps; then
        process_option_apps
        exit $ok
    elif $option_cat; then

```

```

        process_option_cat
        exit $ok
    elif $option_list; then
        process_option_list
        exit $ok
    fi

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

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

    # Search for available APPs in alls setup or reset files, no selection on
    # user name and computer name (USER).
    for file in "$program_path/$program_name-$mode-".*.sh; do
        files+=("$file")
    done

    for app in "${app_arguments[@]}"; do
        file=$(
            grep --files-with-matches \
                --word-regexp \
                --regexp="# APP $app" \
                "${files[@]}" \
                2> >($logcmd) \
            head --lines=1 || printf '\n'
        )
        if [[ -n $file ]]; then
            process_argument_app_file
        else
            file_not_found=true
            printf "$display_name: %s: %s\n" \
                "$app" "$(gettext 'app not found')"
        fi
    done
    if $file_not_found; then
        text=$(eval_gettext "Type '\$display_name --apps' for available apps.")
        printf "%s\n" "$text"
        exit $error
    fi
}

```

```

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

    while read -r record; do
        case $record in
            '')
                # Empty line.
                continue
                ;;
            '# APP '*)
                # App record.
                if [[ $record == '# APP '$app* ]]; then
                    # Found app name.
                    # Write records to cmdsfile until next app name or eof.
                    app_found=true
                    if $first_app_record; then
                        first_app_record=false
                    else
                        printf '\n' >> "$cmdsfile"
                    fi
                    printf '%s\n' "# $file" >> "$cmdsfile"
                    printf '%b\n' "$record" >> "$cmdsfile"
                elif $app_found; then
                    # Next app.
                    app_found=false
                    continue
                fi
                ;;
            '#-APP '*)
                # App hidden record.
                if $app_found; then
                    # Next hidden app.
                    break
                fi
                ;;
            *)
                # Other records.
                if $app_found; then
                    printf '%s\n' "$record" >> "$cmdsfile"
                fi
                ;;
        esac
    done < "$file"
}

```

```

function process_inputfile {
    local apprecord=''
    local first_app_record=true
    local user_match=false
    local record=''

    printf '%s\n\n' "# $inputfile" >> "$cmdsfile"
    while read -r record; do
        case $record in
            '')
                # Empty line.
                continue
                ;;
            '# APP '*|'#-APP '*)
                # App (hidden) record.
                user_match=false

```

```

        apprecord=$record
        ;;
    '# USER '*'*@"*" | '# USER "*" "$USER"@" "$HOSTNAME"*)
        # user record.
        user_match=true
        if $first_app_record; then
            first_app_record=false
        else
            printf '\n' >> "$cmdsfile"
        fi
        {
            printf '%s\n' "$apprecord"
            printf '%s\n' "$record"
        } >> "$cmdsfile"
        ;;
    *)
        # Other records.
        if $user_match; then
            printf '%s\n' "$record" >> "$cmdsfile"
        fi
        ;;
esac
done < "$inputfile"
}

```

```

function process_option_apps {
    local -a files=()

    # Search for available APPs in all setup or reset files.
    for file in "$program_path/$program_name-$mode-"*.sh; do
        files+=("$file")
    done
    text="$(gettext 'The following APPs are available:

    APP')
    $(
        if ! grep --no-messages \
            --regexp="# APP" \
            "${files[@]}" \
            cut --delimiter=' ' \
            --fields=3 \
            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.')"
        fi
    )
    $usage_line"
    else
        printf '%s\n' "
$(gettext 'To setup the APPs run:') ${blue}$display_name APP...${normal}
$(gettext 'To simulate resetting APPs execute:') ${blue}$display_name \
--simulate APP...${normal}
$(gettext 'To view the contents of the APPs setup file execute:') \
${blue}$display_name --cat APP...${normal}"
    fi
)"
    info "$text"
}

```

```

function process_option_cat {
    less --quit-if-one-screen "$cmdsfile"
}

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

    FILE')
$(
    if !    find    "$program_path/$program_name-*.sh"    \
            2> >($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' "
$(eval_gettext "To process FILE execute: \${blue}\$display_name \
--file FILE\${normal}")
$(eval_gettext "To view the commands of FILE execute: \${blue}\$display_name \
--cat --file FILE\${normal}")
$(eval_gettext "To simulate the processing of FILE execute: \
\${blue}\$display_name --simulate --file FILE\${normal}")"
        fi
)"
    info "$text"
}

function process_cmdsfile {
    local app_name=''
    local first_app_line=true
    local operation
    operation=$(gettext 'set up')
    local record=''
    local write_app_line=true
    local -i app_seq_num=0
    local -i app_tot_num=0
    local -i cmd_seq_num=0

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

    if $option_reset; then
        operation=$(gettext 'reset')
    fi
    if $option_simulate; then
        operation+=$(gettext ' simulate')
    fi

    while read -r record; do
        case $record in
            '')
                # Empty line.
                continue

```

```

;;
'# APP '*|'#-APP '* )
# App (hidden) record.
app_name="${record:6} $operation"
write_app_line=true
((++app_seq_num))
if [[ $app_seq_num -gt 99 ]]; then
    app_seq_num=99
fi
cmd_seq_num=0
;;
'# USER '* )
# User record.
continue
;;
'# '* )
# Comment record.
continue
;;
*)
# Other records.
process_command_record "$record"
;;
esac
done < "$cmdsfile"
}

```

```

function process_command_record {
    local cmd=${1:-unknown}
    local app_name_line=''

    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] $app_name\n\n$cmd"
            printf '%s\n' "$text"
        fi
        printf -v app_name_line \
            "[%2d/%-2d] [%2d] %s" \
            "$app_seq_num" \
            "$app_tot_num" \
            "$cmd_seq_num" \
            "$app_name"
        execute_command "$cmd"
    else
        # Option simulate.
        if $write_app_line; then
            if $first_app_line; then
                first_app_line=false
            else
                printf '\n' >> "$simsfile"
            fi
            printf "[%2d/%-2d] %s\n" \
                "$app_seq_num" \
                "$app_tot_num" \
                "$app_name" >> "$simsfile"
        fi
        write_app_line=false
    fi
}

```



```

        printf "%7s [%2d] ${blue}%s${normal}\n" \
            " " \
            "$cmd_seq_num" \
            "$cmd" >> "$simsfile"
    fi
}

function execute_command {
    local cmd=${1:-unknown}
    local -i rc=0

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

    log "$app_name_line"
    log "${blue}$cmd${normal}"
    eval "$cmd" |& $logcmd || rc=$?
    maxrc
    # Extra message to the terminal, other is in the log.
    if ! $option_gui && [[ $rc -gt $ok ]]; then
        info "$(eval_gettext "Program \${program_name} encountered an error.")"
    fi
    log "rc=$rc, maxrc=$maxrc"
}

```

```

function term_script {
    local operation
    operation=$(gettext 'set up')

    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 next screen.")"
            show_log
            exit $error
        else
            if $option_reset; then
                # shellcheck disable=SC2034
                operation=$(gettext 'reset')
            fi
            info "
${green}$(eval_gettext "Apps \${operation} completed.")${normal}"
        fi
    else
        # Option simulate.
        less --quit-if-one-screen --RAW-CONTROL-CHARS "$simsfile"
    fi
    exit $ok
}

```

```

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

```

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

```
main "$@"
```