```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Restore backup.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2007-2023.
##############################################################################

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

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


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

program_name='kz-restore'
program_desc=$(gettext 'Restore backup')
display_name=${program_name/kz-/kz }

usage="$(eval_gettext "Usage: \$display_name [-d|--dry-run] [-g|--gui] \
[-s|--source DIRECTORY]")
                  $options_usage"
help="$(eval_gettext "Usage: \$display_name [OPTION...]")

$program_desc.

$(gettext 'Options:')
  $(gettext "Mandatory arguments to long options are mandatory for short \
options too.")

  -d, --dry-run  $(gettext 'perform a test run without making any changes')
  -g, --gui      $(gettext 'starts in graphics mode')
$(gettext '  -s, --source DIRECTORY
                 use backup in DIRECTORY')
$options_help"
options_short+='dgs:'
options_long+=',dry-run,gui,source:'

dry_run_option=''
medium=''
option_dry_run=false
option_gui=false
option_source=false
source_argument=''
source_medium=''
source=''


##############################################################################
# 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
            --dry-run)
                option_dry_run=true
                dry_run_option='--dry-run'
                shift
                ;;
            -g|--gui)
                option_gui=true
                reset_terminal_attributes
                shift
                ;;
            -s|--source)
                if $option_source; then
                    printf  "$display_name: $1 $2: %s\n$usage_line\n" \
                            "$(gettext 'too many options')"
                    exit $error
                fi
                option_source=true
                source_argument=$2
                shift 2
                ;;
            --)
                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 {
    if $option_source; then
        if ! [[ -d $source_argument ]]; then
            printf  "$display_name: $source_argument: %s\n$usage_line\n" \
                    "$(gettext 'directory does not exist')"
            exit $error
        fi
        source=$source_argument/backup-$HOSTNAME-$USER
        source_medium=$source_argument
    else
        medium=$(ls --directory /media/"$USER"/* 2> >($logcmd) || true)
        if [[ -z $medium ]]; then
```

```
        text="$(eval_gettext "No USB medium found.

Connect a USB medium.")"
            warning "$text"
            wait_for_enter
            medium=$(ls --directory /media/"$USER"/* 2> >($logcmd) || true)
            if [[ -z $medium ]]; then
                error "$text"
            exit $error
            fi
        fi
        if [[ $(printf '%s\n' "$medium" | wc --lines) -gt 1 ]]; then
            text="$(eval_gettext "Connect only one USB medium.

Now connected:
\$medium

Disconnect media via Files.")"
            warning "$text"
            wait_for_enter
            if [[ $(printf '%s\n' "$medium" | wc --lines) -gt 1 ]]; then
               error "$text"
                exit $error
            fi
        fi
        source=$medium/backup-$HOSTNAME-$USER
        source_medium=$medium
        if ! [[ -d $source ]]; then
            error "$(eval_gettext "No backup found on connected USB medium.

Connect a USB medium with the directory backup-\$HOSTNAME-\$USER.")"
            exit $error
        fi
    fi

    check_on_ac_power
    restore_backup
    if ! $option_dry_run; then
        title=$(gettext 'Restore settings')
        text=$title
        if $option_gui; then
            restore_settings               |
            zenity  --progress             \
                    --pulsate              \
                    --auto-close           \
                    --no-cancel            \
                    --width     600        \
                    --height    50         \
                    --title     "$title"   \
                    --text      "$text"    2> >($logcmd)
        else
            info "$text..."
            restore_settings
        fi
    fi
}


function restore_backup {
    local -i rc=0
    local target=$HOME

    title=$(gettext 'Restore backup')
    text=$(gettext 'Preparing restore (this can take a while)')
```

```
    if $option_gui; then
        rsync   --archive               \
                --verbose               \
                $dry_run_option         \
                "$source"/              \
                "$target"/              \
                2> >($logcmd)           |
        sed     --expression='s/^/#/'   |
        zenity  --progress              \
                --auto-close            \
                --no-cancel             \
                --pulsate               \
                --width     600         \
                --height    50          \
                --title     "$title"    \
                --text      "$text"     2> >($logcmd) || rc=$?
    else
        info "$text..."
        rsync   --archive               \
                --verbose               \
                --human-readable        \
                $dry_run_option         \
                "$source"/              \
                "$target"/              2> >($logcmd) || rc=$?
    fi
    maxrc

    sync |& $logcmd || rc+=$?
    maxrc
}


function restore_settings {
    local tgtdir
    tgtdir=$HOME/$(gettext 'Settings')

    restore_settings_desktop_background
    restore_settings_favorite_apps

    # Old directory, lagacy.
    rm --force --recursive "$HOME"/kz-data/
}


function restore_settings_desktop_background {
    local file
    file=$tgtdir/$(gettext 'Background')

    if [[ -f $file ]]; then
        gsettings set org.gnome.desktop.background picture-uri "file://$file"
    fi
}


function restore_settings_favorite_apps {
    local file
    file=$tgtdir/$(gettext 'Favorites')

    if [[ -f $file ]]; then
        gsettings set org.gnome.shell favorite-apps "$(cat "$file")"
    fi
}
```

```
function term_script {
    local dev=''
    local -i rc=0

    if $option_dry_run; then
        error "$(gettext 'The backup was NOT restored (DRY RUN).')"
        exit $error
    fi

    if [[ $maxrc -gt $ok ]]; then
        error "$(eval_gettext "Restore ended with warnings or errors.

Some files may not be readable,
or were any files added or removed while restoring the backup,
or is there not enough space on \$target.
The maximum exit value is \$maxrc.
Check the log in the next screen.")"
        show_log
        exit $error
    fi

    if [[ $source_medium == /media/* ]]; then
        term_script_unmount
    else
        info "
${green}$(gettext 'The backup has been restored.')${normal}"
    fi
    exit $ok
}


function term_script_unmount {
    local dev=''
    local -i rc=0

    text=$(gettext 'Disconnect the USB media')
    dev='/dev/'$(
        lsblk   --ascii             |
        grep    --before-context=1  \
                "$source_medium"    |
        head    -1                  |
        cut     --delimiter='-'     \
                --fields=2          |
        cut     --delimiter=' '     \
                --fields=1
    )
    if $option_gui; then
        umount "$source_medium"                     |& $logcmd || rc+=$?;   \
        udisksctl lock --block-device "$dev"        |& $logcmd || rc+=$?;   \
        udisksctl power-off --block-device "$dev"   |& $logcmd             |
        zenity  --progress                                                  \
                --pulsate                                                   \
                --auto-close                                                \
                --no-cancel                                                 \
                --width     600                                             \
                --height    50                                              \
                --title     "$title"                                        \
                --text      "$text"                                         \
                2> >($logcmd) || rc=$?
    else
        info "$text..."
        umount "$source_medium"                     |& $logcmd || rc+=$?
        udisksctl lock --block-device "$dev"        |& $logcmd || rc+=$?
        udisksctl power-off --block-device "$dev"   |& $logcmd || rc+=$?
```

```
    fi
    if [[ $rc -eq $ok ]]; then
        info "
${green}$(gettext 'The backup has been restored.')

$(gettext 'The USB medium can be removed.')${normal}"
    else
            warning "
${green}$(gettext 'The backup has been restored.')${yellow}

$(gettext 'Disconnect the USB medium yourself (safely!).')"
    fi
}


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

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

main "$@"
```