```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Restore backup.
#
# Written in 2007 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-restore'
declare program_desc
program_desc=$(gettext 'Restore backup')
declare display_name=${program_name/kz-/kz }

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

$program_desc.

$(gettext 'Options:')
      --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"
declare options_short+='gs:'
declare options_long+=',dry-run,gui,source:'


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

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


##############################################################################
# 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
            --dry-run)
                option_dry_run=true
                dry_run_option='--dry-run'
                shift
                ;;
            -g|--gui)
                option_gui=true
                kz_common.reset_terminal_attributes
                shift
                ;;
            -s|--source)
                option_source=true
                source_argument=$2
                shift 2
                ;;
            --)
                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 {
    # shellcheck disable=SC2034
    local operation=backup
    local restore=false

    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 $medium ]]; then
             warning "$(gettext "Make sure that the USB medium containing the \
backup is connected.")"
            kz_common.wait_for_enter
            medium=$(ls --directory /media/"$USER"/* 2> >($logcmd) || true)
            if [[ -z $medium ]]; then
                 error "$(eval_gettext "No USB medium found.

Connect a USB medium, and then start \$operation again.")"
            exit $error
            fi
        fi
        if [[ $(printf '%s\n' "$medium" | wc --lines) -gt 1 ]]; then
             error "$(eval_gettext "Connect only one USB medium.

Now connected:
\$medium

Disconnect media via Files, then restart \$operation.")"
            exit $error
        fi
        source=$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, and then \
start restore again.")"
            exit $error
        fi
    fi

    kz_common.check_on_ac_power

    title=$(gettext 'Restore backup')
    text="$(gettext 'Backup copy will be RESTORED.')
"
    if $option_gui; then
        text+="
$(gettext 'Proceed?')"
        if zenity   --question                          \
                    --no-markup                         \
                    --width         600                 \
                    --height        50                  \
                    --title         "$title"            \
                    --text          "$text"             \
                    --ok-label      "$(gettext 'Yes')"  \
                    --cancel-label  "$(gettext 'No')"   2> >($logcmd); then
            restore=true
        else
            restore=false
        fi
    else
        info "$text"
        while true; do
            read -rp "$(gettext 'Continue? [y/N]: ')"
            case $REPLY in
                y*|Y*|j*|J*)
                    restore=true
                    break
                    ;;
                n*|N*|'')
                    restore=false
                    break
```

```
                ;;
            *)
                printf '%s' "${rewrite_line}"
                continue
                ;;
        esac
    done
fi
if $restore; then
    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
else
    exit $ok
fi
}


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

    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> >($logcmd) || rsync_rc=$?
    else
        info "$text..."
        rsync   --archive               \
                --verbose               \
                --human-readable        \
                $dry_run_option         \
                "$source"/              \
                "$target"/              2> >($logcmd) || rsync_rc=$?
    fi
```

```
    case $rsync_rc in
        0)
            return $ok
            ;;
        *)
             error "$(eval_gettext "Restore ended with 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.")"
            exit $error
            ;;
    esac
}


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

    restore_settings_desktop_background
    restore_settings_favorite_apps

    # Old folder, 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

    sync |& $logcmd || rc+=$?
    text=''
    if $option_dry_run; then
         error "$(gettext 'The backup was NOT restored (DRY RUN).')"
        exit $error
    fi
    if [[ $source_medium == /media/* ]]; then
        text=$(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";                    \
            udisksctl lock --block-device "$dev";       \
            udisksctl power-off --block-device "$dev"   |
            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 "
$(gettext 'The backup has been restored.')

$(gettext 'The USB medium can be removed.')"
        else
             warning "
$(gettext 'The backup has been restored.')

$(gettext 'Disconnect the USB medium yourself (safely!).')"
        fi
    else
        info "
$(gettext 'The backup has been restored.')"
    fi
    exit $ok
}


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

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

main "$@"
```