

```

#!/bin/bash
# shellcheck source=kz_common.sh
#####
# Create 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-backup'
declare program_desc
program_desc=$(gettext 'Create backup')
declare display_name=${program_name/kz-/kz }

declare usage
usage="$(eval_gettext "Usage: \${display_name} [-d|--dry-run] \
[-g|--gui] [-t|--target=DIRECTORY]"
        $options_usage)"

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

$program_desc.

$(gettext 'Options:')
  -d, --dry-run  $(gettext 'perform a test run without making any changes')
  -g, --gui      $(gettext 'starts in graphics mode')
$(gettext '  -t, --target=DIRECTORY
        put backup in DIRECTORY')

$options_help"
declare options_short+='dgt:'
declare options_long+=',dry-run,gui,target:'

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

declare dry_run_option=''
declare medium=''
declare option_dry_run=false
declare option_gui=false
declare option_target=false
declare target_argument=''
declare target_medium=''
declare target=''

#####
# 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
        -d|--dry-run)
            option_dry_run=true
            dry_run_option='--dry-run'
            shift
            ;;
        -g|--gui)
            option_gui=true
            kz_common.reset_terminal_attributes
            shift
            ;;
        -t|--target)
            option_target=true
            target_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

    if $option_target; then
        if ! [[ -d $target_argument ]]; then
            printf "$display_name: $target_argument: %s\n$usage_line\n" \
                   "$(gettext 'directory does not exist')"
            exit $error
        fi
        target=$target_argument/backup-$HOSTNAME-$USER
        target_medium=$target_argument
    else
        medium=$(ls --directory /media/"$USER"/* 2> >($logcmd) || true)
        if [[ -z $medium ]]; then

```

```

        text=$(gettext "Connect the USB medium on which the backup will \
be placed.")
        warning "$text"
        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
        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 $error
        fi
        target=$medium/backup-$HOSTNAME-$USER
        target_medium=$medium
    fi

    kz_common.check_on_ac_power

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

function backup_settings {
    local tgt_dir
    tgt_dir=$HOME/$(gettext 'Settings')
    # Settings.

    mkdir --parents "$tgt_dir" |& $logcmd
    backup_settings_desktop_background
    backup_settings_favorite_apps
    backup_settings_installed_apps
    backup_settings_user_photo

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

```

```

function backup_settings_desktop_background {
    local file
    file=$tgtmdir/$(gettext 'Background')
    local picture_file=''

    picture_file=$(
        gsettings get org.gnome.desktop.background picture-uri |
        sed --expression='s|%20| |g' |
        sed --expression="s'/'//g" |
        sed --expression='s|file:///||' || true
    )
    cp --update "$picture_file" "$file" |& $logcmd || true
}

function backup_settings_favorite_apps {
    local file
    file=$tgtmdir/$(gettext 'Favorites')

    gsettings get org.gnome.shell favorite-apps > "$file" |& $logcmd || true
}

function backup_settings_installed_apps {
    local file=$tgtmdir/Apps
    local header1
    header1="$(gettext "[1/3] These packages are installed via Add or Remove \
Programs, with a command\n
such as 'sudo apt --install <package|file>', or with kz install:")"
    local header2
    header2="$(gettext "[2/3] These snaps are installed via Add or Remove \
Programs,\n
with '[sudo] install <snap>', or with kz install:")"
    local header3
    header3="$(gettext "[3/3] These repositories are added with the \
installation of a package, with\n
the command 'sudo add-apt-repository ppa:<ppa-user>/<ppa-name>', or\n
with kz install:")"
    local search=/etc/apt/sources.list

    printf '%b\n' "$header1" > "$file"
    if ! apt list --manual-installed 2> >($logcmd) >> "$file"; then
        true
    fi

    printf '\n%b\n' "$header2" >> "$file"
    if ! snap list 2> >($logcmd) >> "$file"; then
        printf '%s\n' "$(gettext 'snap is not installed')" >> "$file"
    fi

    printf '\n%b\n' "$header3" >> "$file"
    cd $search.d
    if ! grep --recursive \
        --no-filename \
        --regexp='^deb ' \
        --recursive \
        /etc/apt/sources.list \
        /etc/apt/sources.list.d |
        grep --invert-match \
        --regexp='ubuntu.com' |
        sort --unique >> "$file"; then
        true
    fi
}

```

```

}

function backup_settings_user_photo {
    local srcdir=/var/lib/AccountsService/icons
    local file
    file=$tgt_dir/$(gettext 'Userphoto')

    cp --update $srcdir/"$USER" "$file" |& $logcmd || true
}

function create_backup {
    local exclude=''
    local -i rsync_rc=0
    local source=$HOME

    exclude=$(mktemp -t "$program_name-XXXXXXXXXX")

    # Déjà Dup: Some locations are ignored by default.
    cat << EOF > "$exclude"
.adobe/Flash_Player/AssetCache
.cache
.ccache
.gvfs
.Private
.recent-applications.xbel
.recently-used.xbel
.steam/root
.thumbnails
.var/app/*/cache
.xsession-errors
snap/**/.cache
*CACHEDIR.TAG*
EOF

    # Skip git-controlled directories.
    cat << EOF >> "$exclude"
$(
    find    "$HOME"          \
           -maxdepth 2      \
           -name .git        \
           -type d           \
           -print            |
    cut    --delimiter='/' \
           --fields=4        |
    sort
)
EOF

    title=$(gettext 'Create backup')
    text=$(gettext 'Preparing backup (this may take a while)')
    # title=$(gettext '')
    # text=$(gettext '')
    if $option_gui; then
        rsync --archive          \
              --verbose          \
              $dry_run_option    \
              --delete           \
              --exclude-from="$exclude" \
              --delete-excluded  \
              "$source"/         \
              "$target"/         \
              2> >($logcmd)      |
    fi
}

```

```

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          \
        $dry_run_option           \
        --delete                  \
        --exclude-from="$exclude" \
        --delete-excluded         \
        "$source"/               \
        "$target"/               2> >($logcmd) || rsync_rc=$?
fi
rm "$exclude"

case $rsync_rc in
  0)
    return $ok
    ;;
  *)
    error "$(eval_gettext "Backup ended with errors.

Some files may not be readable,
or were any files added or removed while creating the backup,
or is there not enough space on \${target}.")"
    exit $error
    ;;
esac
}

```

```

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

  text=$(gettext "Writing data from temporary memory (this may take a \
while)")
  if ! $option_dry_run; then
    if $option_gui; then
      sync |
      zenity --progress           \
            --pulsate            \
            --auto-close         \
            --no-cancel          \
            --width      600     \
            --height     50     \
            --title      "$title" \
            --text       "$text" 2> >($logcmd) || rc+=$?
    else
      info "
$text..."
      sync |& $logcmd || rc+=$?
    fi
  fi
}

```

```

text=''
if $option_dry_run; then
    error "$(gettext 'The backup has NOT been created (DRY RUN).')"
    exit $error
fi
if [[ $target_medium == /media/* ]]; then
    text=$(gettext 'Disconnect the USB media')
    dev='/dev/'$(
        lsblk --ascii |
        grep --before-context=1 \"$target_medium\" |
        head -1 |
        cut --delimiter='-' \\
        --fields=2 |
        cut --delimiter=' ' \\
        --fields=1
    )
    if $option_gui; then
        umount "$target_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 "$text..."
        umount "$target_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 made.')

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

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

```

```

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

```

```

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

```

```
main "$@"
```