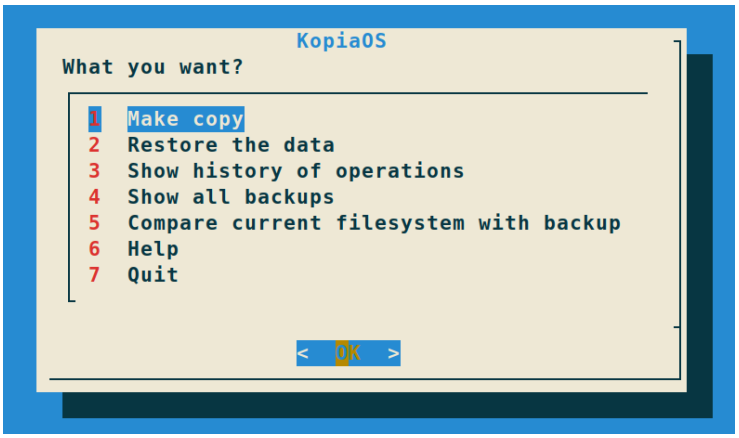
Skrypt wspomagający sporządzenie kopii zapasowej KopiaOS by Dawid Bednarczyk

Skrypt został napisany w powłoce BASH z użyciem pakietu DIALOG, całość została napisana w edytorze Emacs w wersji terminalowej.

ZALECANE: skrypt przechowywać w folderze dla niego stworzonym, przez wzgląd na tworzenie plików tymczasowych przez skrypt jak I stałych folderów z zawartością backup'ów. Nie należy usuwać plików/katalogów stworzonych przez skrypt!!! Pliki tymczasowe są usuwane automatycznie!!!

Ekran powitalny:

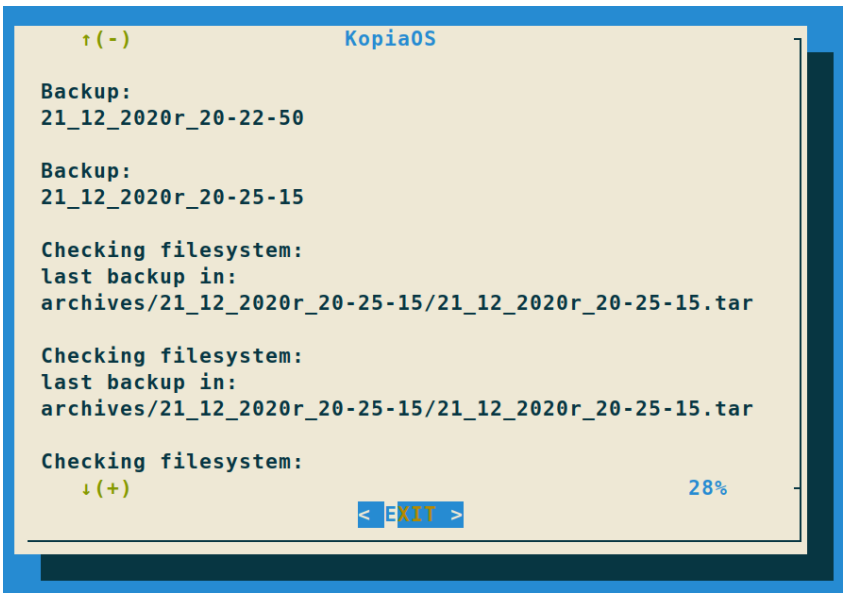
Funkcjonalności skryptu:



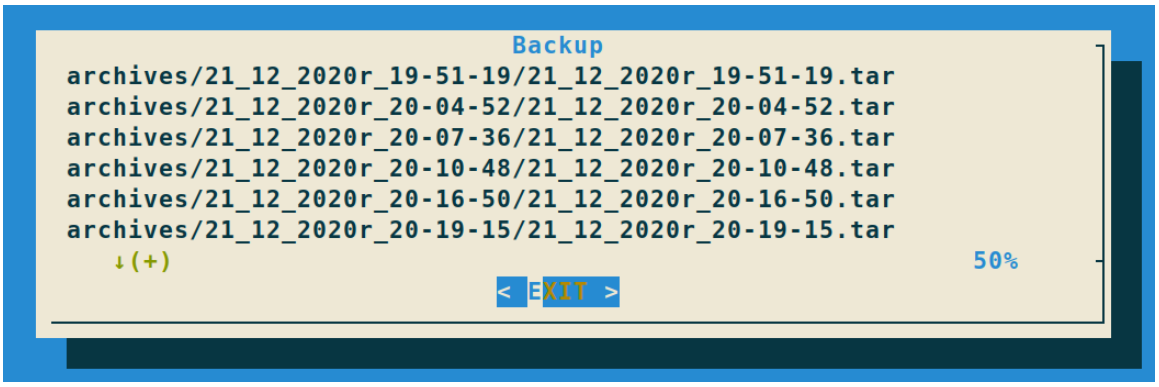
Make copy – funkcja wspomagająca tworzenie kopii zapasowej wybranej części systemu plików, posiadająca zabezpieczenia uniemożliwiające zachowania takie jak: dodawanie podfolderów/plików znajdujących się w danym folderze, dodawanie folderów będących folderami nadrzędnymi danych folderów/plików

Resore the data – umożliwia odzyskanie danych na dwa sposoby: odzyskanie brakujących plików lub całkowite odzyskanie danych z danego archiwum.

Show history of operations – pokazuje historie przeprowadzonych operacji

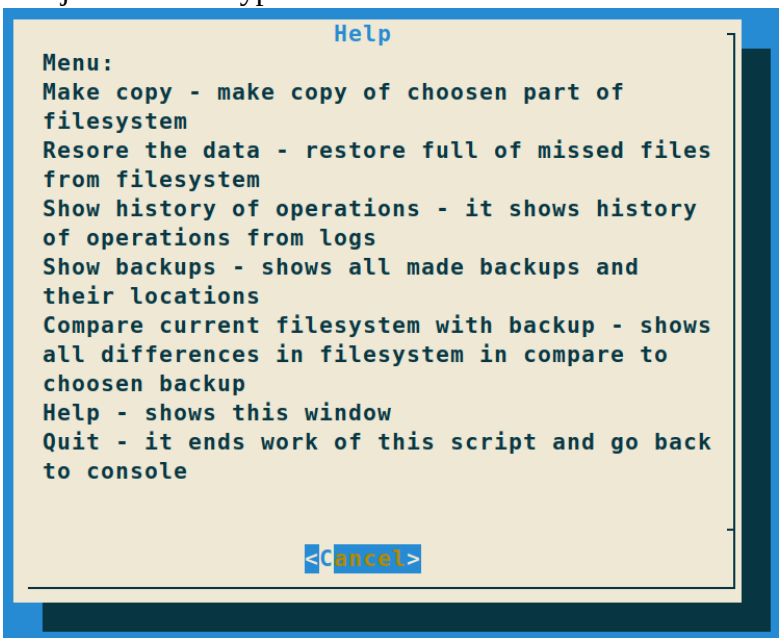


Show all backups – pokazuje wszystkie backupy



Compare current filesystem with backup – porównuje dany system plikowy z daną kopią

Help – krótki opis funkcjonalności skryptu

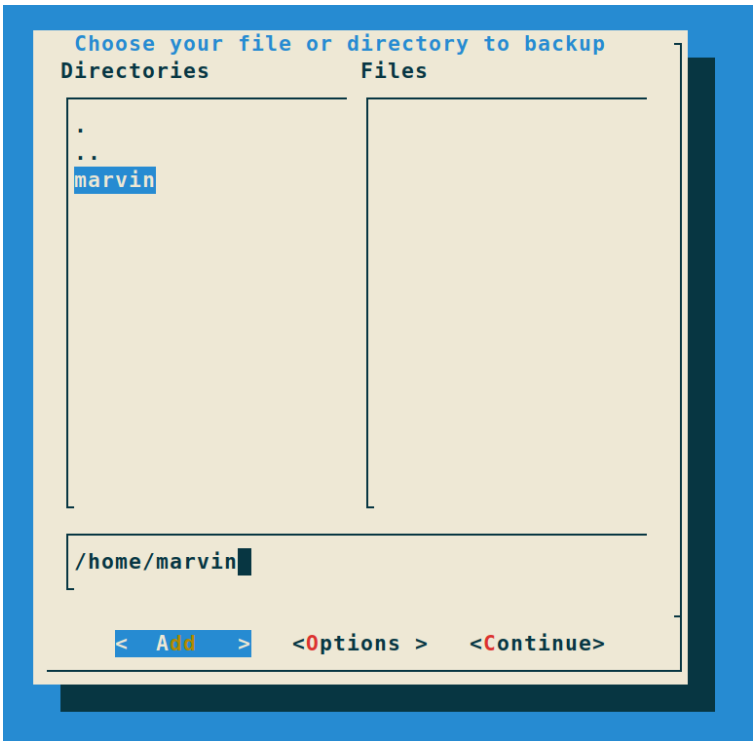


Omówienie bardziej złożonych funkcjonalności:

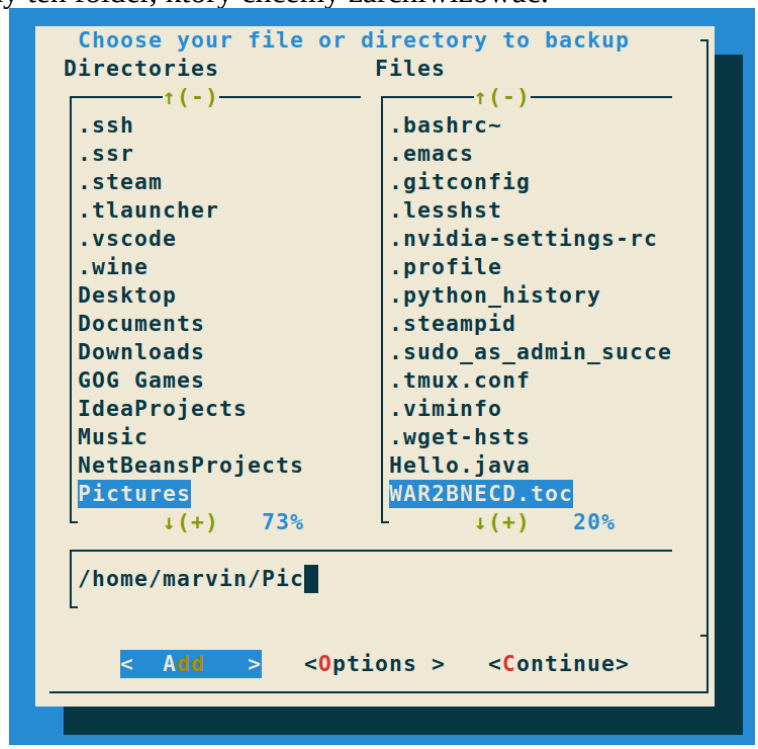
1. Make copy
2. Restore the data
3. Compare current filesystem with backup

Make copy

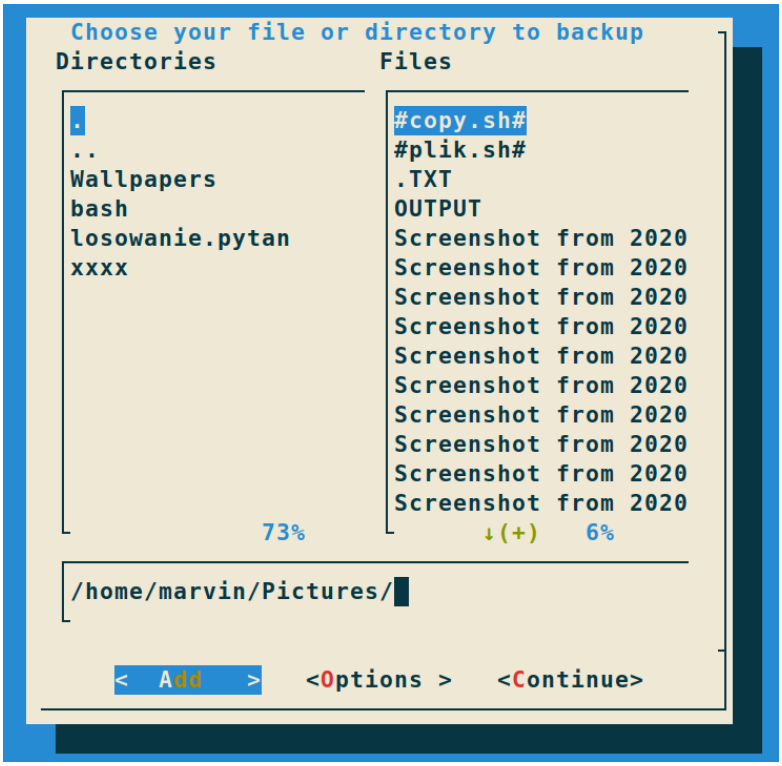
Po wybraniu tej opcji ukaże się nam okno dialogowe do wyboru plików I folderów do zarchwizowania:



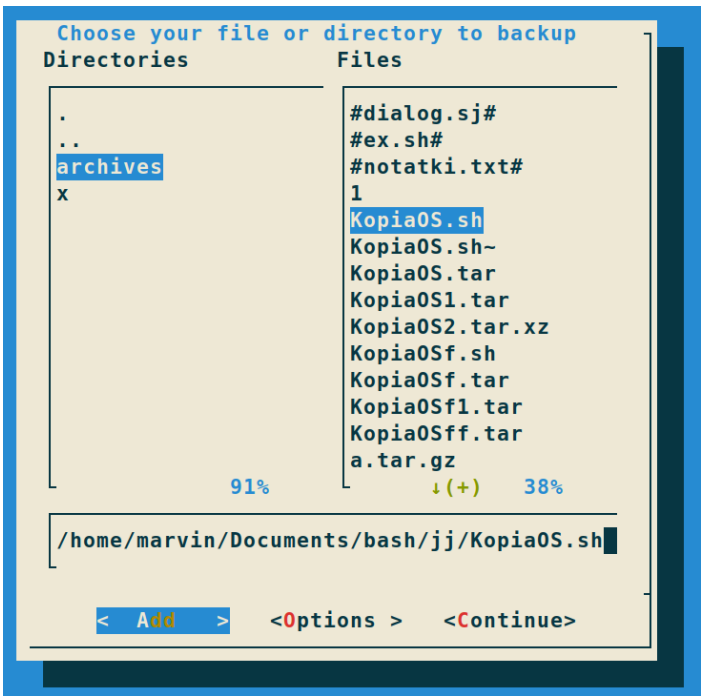
Następnie wybieramy ten folder, który chcemy zarchiwizować:



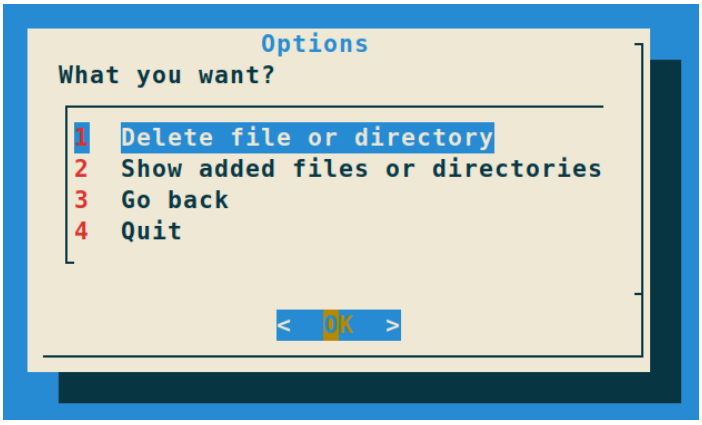
W przypadku pasującego wzorca wystarczy wcisnąć spację z nazwa się sama uzupełni:

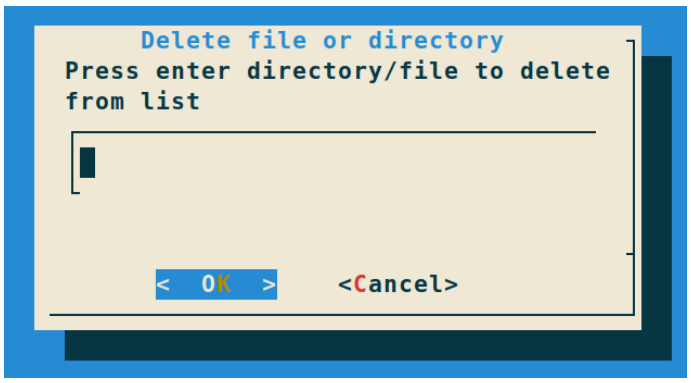


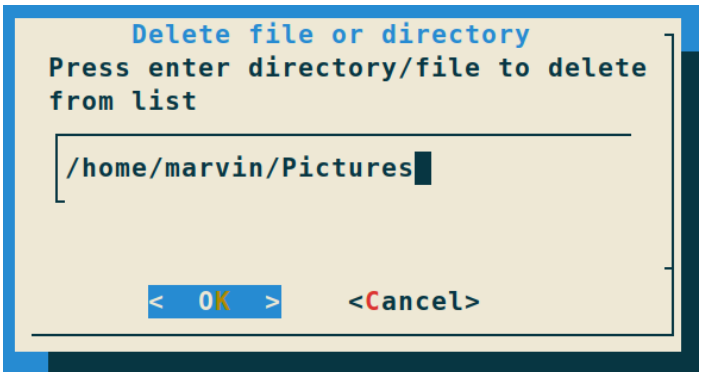
Możemy także wybrać dowolny plik/i pod warunkiem że nie znajduje się w ramach dodanego wcześniej katalogu:



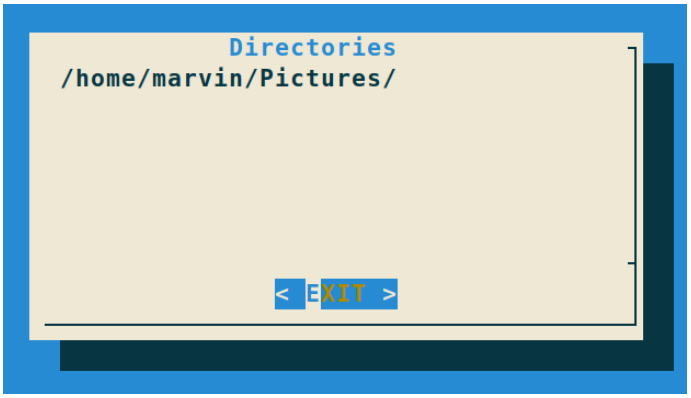
Możemy w opcjach usunąć dany plik lub katalog pod warunkiem że został on wcześniej dodany, sprawdzić co dodaliśmy, cofnąć I wyjść ze skryptu:

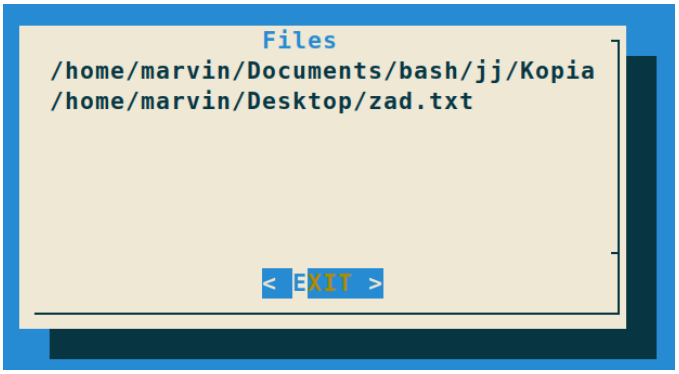



Delete file or directory
Press enter directory/file to delete from list
< OK >
<Cancel>


Delete file or directory
Press enter directory/file to delete from list
/home/marvin/Pictures
< OK >
<Cancel>


/home/marvin/Pictures deleted
< Next >

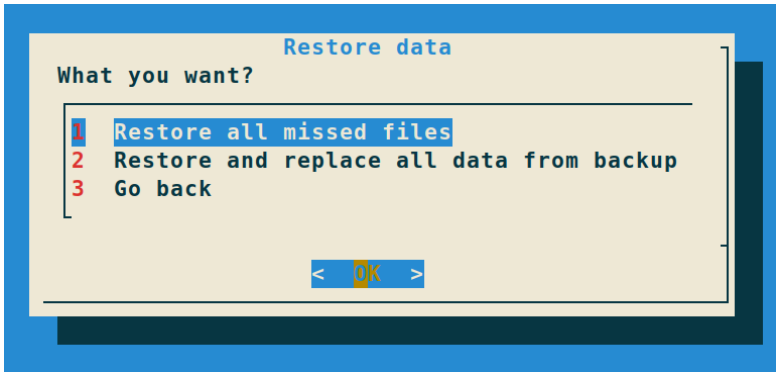

Directories
/home/marvin/Pictures/
< EXIT >



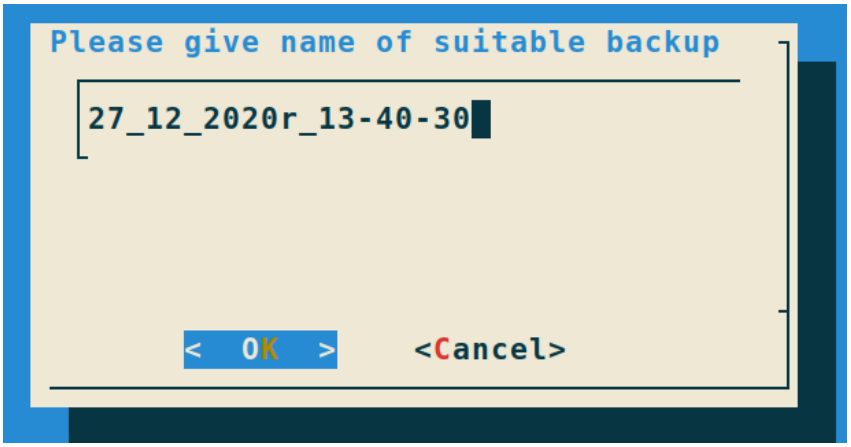
Jeśli jesteśmy gotowi to możemy stworzyć archiwum poprzez wybranie continue, program zapyta czy chcemy kontynuować. Po wykonaniu archiwum zostanie wyświetlony czas w jakim zostało wykonane:

Restore the data

Wybieramy interesującą nas opcję:

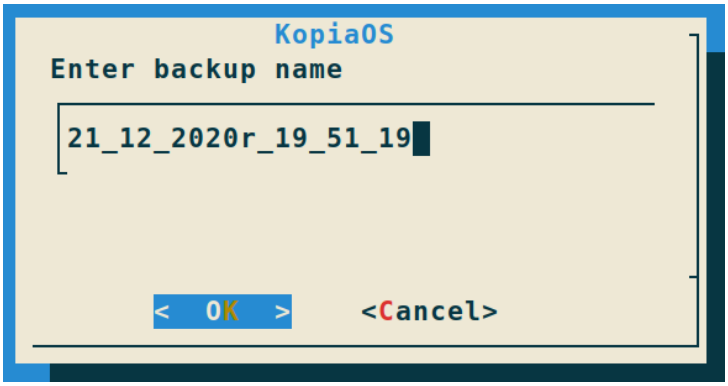


Wprowadzamy nazwę archiwum z którego chcemy skorzystać I o ile takie istnieje I przywracamy system plików:

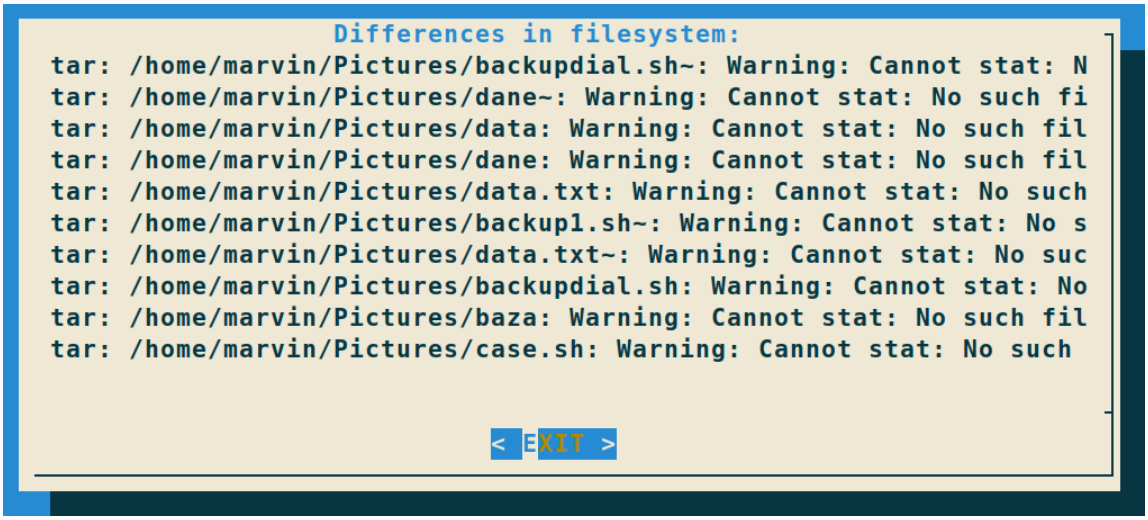


Compare current filesystem with backup

Funkcja porównująca system plików z danym tarem:



Wynik zwracany w osobnym oknie (może się zdarzyć, że niektóre linie zwróconego wyniku będą dłuższe niż długość okna, można nawigować strzałkami by zobaczyć resztę tekstu):



Kod:

```
#!/bin/bash

if [ ! $(which dialog) ]; then
   clear
   RED='\033[0;31m'
   NC='\033[0m'
   echo -ne "${RED}This script requires dialog package!${NC}"
   echo -ne ""
   exit
fi

function main(){
   OUTPUT="temp"
   >$OUTPUT
   title="KopiaOS"
   story="history.txt"
   backtitle="KopiaOS by Dawid Bednarczyk"

   dialog --clear --title "$title" --backtitle "$backtitle" --ok-label "Next" --msgbox "\nHello in
KopiaOS!" 7 30

   BUTTON=0
   while true; do
         deleteTrash
         menu
         BUTTON=$?
         case $choice in
            1) makecopy;;
            2) restoreData;;
            3) operations;;
            4) showbackup;;
            5) compareWithFilesystem;;
            6) Help;;
            7) clear
               exit;;
            *) continue;;
         esac
   done
   clear
}

function menu(){
   choice=$(dialog --clear --title "$title" --backtitle "$backtitle" --no-cancel --menu "What you
want?" 14 50 7 1 "Make copy" 2 "Restore the data" 3 "Show history of operations" 4 "Show all
backups" 5 "Compare current filesystem with backup" 6 "Help" 7 "Quit" 2>&1 > /dev/tty)
   return
}


function comp(){
```

```
    local str1=$1
    local str2=$2
    local len=${#str1}
    if [ "$str1${str2:$len}" == "$str2" ]; then
        echo "$str1${str2:$len}" >> temp.txt
        return 0
    else
        return 1
    fi
}


function operations(){
    if [[ ! -f "$story" ]]; then
        $(touch "$story")
    fi
    dialog --clear --title "$title" --backtitle "$backtitle" --textbox "$story" 20 60
    return
}


function deleteTrash(){
    if [ -f $dirs ]; then
        rm $dirs
    fi
    if [ -f $files ]; then
        rm $files
    fi
    if [ -f copy.txt ]; then
        rm copy.txt
    fi
    if [ -f dirstemp.txt ]; then
        rm dirstemp.txt
    fi
    if [ -f temp.txt ]; then
        rm temp.txt
    fi
    if [ -f ans.txt ]; then
        rm ans.txt
    fi
    if [ -f ddirs.txt ];then
        rm ddirs.txt
    fi
    if [ -f delpth ]; then
        rm "delpth"
    fi
    if [ -f dirsduplicate.txt ]; then
        rm dirsduplicate.txt
    fi
    if [ -f temp ]; then
        rm "temp"
    fi
```

```
    if [ -f summary.txt ]; then
        rm summary.txt
    fi
    if [ -f time.txt ]; then
        rm time.txt
    fi
    if [ -f compatibility.txt ];then
        rm compatibility.txt
    fi

}


function makecopy(){
    files="filestocopy.txt"
    dirs="directoriestocopy.txt"

    deleteTrash

    $(touch $files)
    $(touch $dirs)

    while true ; do
        FD=$(dialog --clear --title "Choose your file or directory to backup" --stdout --backtitle
"$backtitle" --ok-label "Add" --extra-button --extra-label "Options" --cancel-label "Continue" --
fselect $HOME 14 48)
        BUTTON=$?
        if [ "$BUTTON" -eq 0 ]; then
            if [ ! -z "$FD" ]; then

                flag=0

                checkDirectory
                checkFile

            fi

        elif [ "$BUTTON" -eq 3 ]; then
            options
        elif [ "$BUTTON" -eq 1 ]; then
            cat $dirs > summary.txt
            cat $files >> summary.txt
            dialog --clear --title "Are you sure to continue?" --backtitle "$backtitle" --ok-label "Yes"
--extra-button --extra-label "No" --textbox summary.txt 10 40
            if [ $? -eq 0 ]; then
                { time makeTar ; } 2> time.txt
                dialog --clear --title "Backup $toDate is ready!" --backtitle "$backtitle" --ok-label
"Next" --textbox time.txt 10 60
                deleteTrash
                return
            fi
        fi
```

```
    done
}


function checkDirectory(){
    ####if dla katalogu
    if [ -d "$FD" ]; then
        while read line; do
            if [ "$line" == "$FD" ] || [ "$line/" == "$FD" ]; then
                dialog --clear --title "Important!" --backtitle "$backtitle" --msgbox "This directory is
already added to backup!" 10 40
                flag=1
                break
            fi
        done < "$dirs"


        ####sprawdza ile jest podkatalogów
        if [ $flag -eq 0 ]; then
            dupDict=0
            while read line; do
                $(comp "$FD" "$line")
                if [ "$?" -eq 0 ]; then
                    grep "$FD" "$dirs" > dirsduplicate.txt
                    #cat dirscopy > $dirs
                    dupDict=$[$dupDict+1]
                fi
            done < "$dirs"
            if [ $dupDict -ne 0 ]; then
                dialog --clear --extra-button --extra-label "No" --ok-label "Yes" --title "This
subdirectories are in $FD. Do you want delete them?" --backtitle "$backtitle" --textbox
dirsduplicate.txt 5 70
                if [ "$?" -eq 0 ]; then
                    while read line; do
                        grep -v $line $dirs > ddirs.txt
                        cat ddirs.txt > $dirs
                    done <dirsduplicate.txt
                else
                    flag=1
                fi
            fi
            ####koniec sprawdzania podkatalogów


            ####sprawdzanie czy folder nie jest podkatalogiem
        elif [ $flag -eq 0 ]; then
            while read line; do
                $(comp $line $FD)
                #echo $? >> ans.txt
                if [ "$?" -eq 0 ]; then
```

```
                    dialog --clear --extra-button --extra-label "No" --ok-label "Yes" --title "This is subdirectory of $line. Do you want delete it and add?" --backtitle "$backtitle" --textbox dirsduplicate.txt 10 70
                    if [ "$?" -eq 0 ]; then
                        grep -v "$line" $dirs > ddirs.txt
                        cat ddirs.txt > $dirs
                        break
                    else
                        flag=1
                        break
                    fi
                fi
            done <$dirs
        fi
        ####

        count=$((grep "$FD" "$files") | (wc -l))

        ####sprawdza ile plików w danym katalogu
        if [ $count -gt 0 ] && [ $flag -eq 0 ]; then
            dialog --clear --title "Message" --backtitle "$backtitle" --yesno "This folder contains many files added to back up in count $count.\nPress no to cancel this action, but this directory will not be added to backup!\nIf you choose yes, then directory will be added but all files which are there will be remove from list to backup due of collisions during backup!" 5 70
            if [ "$?" -eq 0 ];then
                grep -v "$FD" $files > filescopy.txt
                cat filescopy.txt > $files
            else
                flag=1
            fi
        fi

        if [ $flag -ne 1 ]; then
            if [ "${FD: -1}" != "/" ]; then
                FD="$FD/"
            fi
            echo "$FD" >> "$dirs"
            dialog --clear --title "Message" --backtitle "$backtitle" --msgbox "Directory added!" 10 40
        fi

    fi

}


function checkFile(){

    ####if dla pliku
    if [ -f "$FD" ]; then
        flag=0
        while read line; do
```

```
            if [ "$line" == "$FD" ]; then
                dialog --clear --title "Important!" --backtitle "$backtitle" --msgbox "This file is
already added to backup!" 10 40
                flag=1
                break
            fi
        done < "$files"
        if [ $flag -eq 0 ]; then
            while read line; do
                $(comp "$line" "$FD")
                #echo $? > ans.txt
                if [ "$?" -eq 0 ]; then
                    dialog --clear --title "Message" --backtitle "$backtitle" --yesno "This file is in the
directory from list to backup!\nIf you want to add files from $line instead all directory you must
delete this from list.\nDo you want to delete $line from the list?" 10 40
                    flag=1
                    if [ "$?" -eq 0 ]; then
                        grep -v "$line" $dirs > dirstemp.txt
                        cat dirstemp.txt > $dirs
                        flag=0
                        break
                    fi
                fi
            done < "$dirs"
        fi

        if [ $flag -ne 1 ]; then
            echo "$FD" >> "$files"
            dialog --clear --title "Message" --backtitle "$backtitle" --msgbox "File added!" 10 40
        fi
    fi

}


function options(){
    opt=0
    while [ $opt -ne 3 ]; do
        opt=$(dialog --clear --title "Options" --no-cancel --backtitle "$backtitle" --menu "What you
want?" 11 40 4 1 "Delete file or directory" 2 "Show added files or directories" 3 "Go back" 4 "Quit"
2>&1 > /dev/tty)

        if [ $opt -eq 4 ]; then
            exit
        elif [ $opt -eq 2 ];then
            dialog --clear --title "Directories" --backtitle "$backtitle" --textbox "$dirs" 10 40
            dialog --clear --title "Files" --backtitle "$backtitle" --textbox "$files" 10 40
        elif [ $opt -eq 1 ];then
            pth=$(dialog --clear --title "Delete file or directory" --backtitle "$backtitle" --inputbox
"Press enter directory/file to delete from list" 10 40 2>&1 > /dev/tty)
            if [ $? -eq 0 ]; then
                if [ "$pth" == "$(cat $dirs | grep -x "$pth")" ]; then
```

```
                    cat $dirs | grep -v "$pth" > copy.txt
                    cat copy.txt > $dirs
                    dialog --clear --title "" --backtitle "$backtitle" --ok-label "Next" --msgbox "$pth
deleted" 7 30
                elif [ "$pth/" == "$(cat $dirs | grep -x "$pth/")" ]; then
                    cat $dirs | grep -v "$pth/" > copy.txt
                    cat copy.txt > $dirs
                    dialog --clear --title "" --backtitle "$backtitle" --ok-label "Next" --msgbox "$pth
deleted" 7 30
                elif [ "$pth" == "$(cat $files | grep -x "$pth")" ]; then
                    cat $files | grep -v "$pth" > copy.txt
                    cat copy.txt > $files
                    dialog --clear --title "" --backtitle "$backtitle" --ok-label "Next" --msgbox "$pth
deleted" 7 30
                else
                    dialog --clear --title "" --backtitle "$backtitle" --ok-label "Next" --msgbox "No
such file or directory!" 7 30
                fi
            fi
        fi
    done
}


function makeTar(){
    if [ ! -d "archives" ]; then
        mkdir "archives"
    fi
    toDate="$(date '+%d_%m_%Yr_%H-%M-%S')"
    newb="archives/$toDate"
    mkdir $newb
    #tar --create -f "$newb/$toDate".tar $dirs $files
    while read line; do
        dialog --title "Compressing..." --backtitle "$backtitle" --infobox "Compressing following
directory: $line" 10 40
        tar --append -f "$newb/$toDate".tar --absolute-names $line
    done <$dirs
    while read line; do
        dialog --title "Compressing..." --backtitle "$backtitle" --infobox "Compressing following
file: $line" 10 40
        tar --append -f "$newb/$toDate".tar --absolute-names $line
    done <$files
    cp $dirs $newb
    cp $files $newb
    echo "Backup:" >> $story
    echo -e "$toDate" >> $story
    echo "  " >> $story
    echo "$newb/$toDate.tar" >> all_backup.txt
}


function compareWithFilesystem(){
```

```
    if [ -z "$(tail -n 1 all_backup.txt)" ]; then
        dialog --clear --title "$title" --backtitle "$backtitle" --msgbox "There is no last backup!" 10
40
        return
    fi
    dialog --clear --title "$title" --backtitle "$backtitle" --yesno "Do you want use last backup to
compare with filesystem?" 10 40
    if [ $? -eq 0 ]; then
        bup="$(tail -n 1 all_backup.txt)"
        checkTree "last backup" $bup
    else
        bup=$(dialog --clear --title "$title" --backtitle "$backtitle" --inputbox "Enter backup name"
10 40  2>&1 > /dev/tty)
        if [ $? -eq 0 ]; then
            if [ -f "archives/$bup/$bup.tar" ]; then
                checkTree "$bup" "archives/$bup/$bup.tar"
            else
                dialog --clear --title "Message!" --backtitle "$backtitle" --msgbox "There is no file of
backup!" 10 40
            fi
        else
            return
        fi
    fi
}


function checkTree(){
    dialog --clear --title "Cheking..." --backtitle "$backtitle" --infobox "Checking $1 " 10 40
    tar --compare --absolute-names -f $2 2> compatibility.txt
    sleep 3
    dialog --clear --title "Differences in filesystem: " --backtitle "$backtitle" --textbox
compatibility.txt 15 70
    echo "Checking filesystem:" >> $story
    echo "$1 in:" >>$story
    echo "$2" >>$story
    echo "  " >> $story
}


function showbackup(){
    if [ ! -f all_backup.txt ];then
        touch all_backup.txt
    fi

    dialog --clear --title "Backup" --backtitle "$backtitle" --textbox all_backup.txt 10 70
}

function restoreData(){
    while true; do
        if [ $(tail -n 1 all_backup.txt | wc -l) -lt 1 ]; then
```

```
            dialog --clear --title "Error!" --backtitle "$backtitle" --msgbox "There is no any backup!"
10 40
            return
        fi
        rstr=$(dialog --clear --title "Restore data" --backtitle "$backtitle" --no-cancel --menu "What
you want?" 10 50 3 1 "Restore all missed files" 2 "Restore and replace all data from backup" 3 "Go
back" 2>&1 > /dev/tty)
        if [ "$rstr" -eq 3 ]; then
            break
        else
            bpth=$(dialog --clear --title "Please give name of suitable backup" --backtitle "$backtitle"
--inputbox "" 10 40 2>&1 > /dev/tty)
            if [ $? -eq 1 ]; then
                continue
            elif [ -z $bpth ]; then
                dialog --clear --title "Error!" --backtitle "$backtitle" --msgbox "Path is empty!!!!" 10
40
            elif [ "$rstr" -eq 2 ];then
                if [ $(grep $bpth all_backup.txt | wc -l) -eq 1 ]; then
                    touch errors.txt
                    tar --extract --absolute-names -f $(grep $bpth all_backup.txt) -C / 2> errors.txt
                    echo "Restore all data" >> $story
                    echo "from: $bpth" >> $story
                    echo " " >> $story
                    if [ $(cat errors.txt | wc -l) -gt 0 ];then
                        dialog --clear --title "Restoring data ends with errors!" --backtitle "$backtitle"
--textbox errors.txt 20 60
                    else
                        dialog --clear --title "Success!" --backtitle "$backtitle" --msgbox "Data
restored!" 10 40
                    fi
                fi
            elif [ "$rstr" -eq 1 ]; then
                if [ $(grep $bpth all_backup.txt | wc -l) -eq 1 ]; then
                    touch errors.txt
                    tar --extract --absolute-names --keep-old-files -f $(grep $bpth all_backup.txt) -C /
2>errors.txt
                    echo "Restore missed data" >>$story
                    echo "from $bpth" >>$story
                    echo " " >>$story
                    if [ $(cat errors.txt | wc -l) -gt 0 ];then
                        dialog --clear --title "Restoring data ends with errors!" --backtitle "$backtitle"
--textbox errors.txt 20 60
                    else
                        dialog --clear --title "Success!" --backtitle "$backtitle" --msgbox "Data
restored!" 10 40
                    fi
                fi
            fi
        fi
    done
}
```

```
function Help(){
    dialog --clear --title "Help" --backtitle "$backtitle" --ok-label "Cancel" --msgbox "Menu:\nMake copy - make copy of choosen part of filesystem\nResore the data - restore full of missed files from filesystem\nShow history of operations - it shows history of operations from logs\nShow backups - shows all made backups and their locations\nCompare current filesystem with backup - shows all differences in filesystem in compare to choosen backup\nHelp - shows this window\nQuit - it ends work of this script and go back to console" 20 50
}

main
```