

Kamil Breczko

# System plików

18 stycznia 2017

# Spis treści

<b>1. Wprowadzenie</b>	3
1.1. Interfejs systemu plików	3
<b>Struktury katalogów</b>	4
<b>2. Katalog jednopoziomowy</b>	4
2.1. Opis zadania	4
2.2. Opis skryptu	4
2.3. Sposób uruchamiania	4
2.4. Przykład	5
<b>3. Katalog dwupoziomowy</b>	6
3.1. Katalogowanie plików wg daty modyfikacji	6
3.1.1. Opis skryptu	6
3.1.2. Sposób uruchamiania	6
3.1.3. Przykład	6
3.2. Katalogowanie plików wg rozszerzenia	6
3.2.1. Opis skryptu	6
3.2.2. Sposób uruchamiania	7
3.2.3. Przykład	7
<b>4. Drzewiasta struktura katalogów</b>	8
4.1. Struktura grafu acyklicznego	8
4.2. Zapisywanie struktury do pliku	8
4.2.1. Opis skryptu	8
4.2.2. Sposób uruchamiania	9
4.2.3. Przykład	9
4.3. Tworzenie struktury z pliku	9
4.3.1. Opis skryptu	9
4.3.2. Sposób uruchamiania	9
4.3.3. Przykład	10
<b>5. Proste operacje na plikach</b>	11
5.1. Synchronizacja danych	11
5.1.1. Sposób uruchamiania	11
5.1.2. Przykład	11
5.2. Monitorowanie miejsca na dysku	11
5.2.1. Sposób uruchamiania	12
5.3. Usuwanie linków	12
5.3.1. Sposób uruchamiania	12
5.3.2. Przykład	12
<b>6. Podsumowanie</b>	13
<b>Literatura</b>	13

# 1. Wprowadzenie

System plików pełni w systemie operacyjnym rolę pamięci trwałej. Informacje zapisane w pamięci trwałej nie są gubione między kolejnymi uruchomieniami systemu. Zwykle pamięć trwała, to dyski magnetyczne. Informacje przechowywane w systemie plików są pogrupowane w pliki i katalogi.[1]

Wszystkie opisane poniżej skrypty zostały wykonane w powłoce systemowej UNIX.

## 1.1. Interfejs systemu plików

W systemie plików występują dwa podstawowe rodzaje obiektów: pliki i katalogi. Pliki służą bezpośrednio do przechowywania informacji, a katalogi służą do grupowania plików i innych (pod)katalogów. Można powiedzieć, że katalog to pojemnik na pliki (lub inne katalogi), pozwalający je katalogować, zamiast składować bezpośrednio w katalogu głównym systemu plików.

Typowe cechy pliku:

**Nazwa** – symboliczna nazwa identyfikująca plik w obrębie katalogu, w którym się on znajduje. W niektórych systemach plików, ten sam plik może występować pod wieloma nazwami i w wielu katalogach. Nie zawsze więc plik ma jednoznacznie określoną nazwę, natomiast zawsze nazwa i katalog jednoznacznie określają plik.

**Typ** – określa rodzaj informacji przechowywanej w pliku.

**Treść** – informacje przechowywane w pliku. W zależności od rodzaju systemu operacyjnego, treści plików mogą posiadać zróżnicowaną strukturę lub nie.

**Wskaźnik pliku** – wskazuje miejsce w pliku, którego będzie dotyczyć kolejna operacja czytania/pisania.

**Prawa dostępu** – w systemach zapewniających ochronę danych, z każdym plikiem (jak również z katalogiem) są związane prawa dostępu, określające jakie operacje mają prawo wykonywać dani użytkownicy. Dokładny zestaw praw dostępu i ich znaczenie zależy od konkretnego systemu operacyjnego.

**Czas utworzenia/ostatniej modyfikacji/ostatniego dostępu** – są to dane o charakterze administracyjnym, ułatwiające archiwizację, porównywanie wersji plików, zarządzanie plikami tymczasowymi itp.

Wszystkie z tych atrybutów mają również odniesienie do katalogów, pod warunkiem, że za treść katalogu przyjmiemy listę plików i podkatalogów znajdujących się w danym katalogu.

Typowe operacje na plikach/katalogach, to:

- zmiana nazwy pliku/katalogu;
- przeniesienie pliku/katalogu do innego katalogu;
- usunięcie pliku/katalogu;
- skopiowanie pliku;
- utworzenie nowego (pustego) katalogu;
- wypisanie treści pliku/zawartości katalogu;

## Struktury katalogów

Katalogi służą do grupowania plików. Struktury katalogów ewoluowały w miarę rozwoju systemów operacyjnych, odzwierciedlając pojawiające się potrzeby użytkowników.

### 2. Katalog jednopoziomowy

Katalog jednopoziomowy to najprostsza struktura katalogów. W systemie jest tylko jeden katalog, w którym zgromadzone są wszystkie pliki. Każdy plik jest jednoznacznie identyfikowany przez swoją nazwę. Taka prymitywna struktura realizuje jedynie podstawową potrzebę gromadzenia wielu plików w systemie.

#### 2.1. Opis zadania

Skrypt polega na spłaszczeniu podanego na wejściu katalogu, tak aby wszystkie pliki znajdowały się w jednym katalogu.

#### 2.2. Opis skryptu

Skrypt przyjmuje jeden argument, ścieżka do katalogu, która oznacza korzeń drzewa katalogów do spłaszczenia. Posiada jedną zmienną globalną, którą jest głębokość spłaszczenia. Na początku przyjmuje wartość -1, co oznacza spłaszczenie całej struktury katalogów. Operacja samego tworzenia struktury katalogu jednopoziomowego wygląda następująco — dla każdego podkatalogu w drzewie katalogu spłaszczanego, obojętnie na którym poziomie, zostaną wybrane wszystkie pliki znajdujące się w podanej lokalizacji i przeniesione do katalogu głównego z parametrem informującym czy dana nazwa pliku już istnieje. Jeśli program napotka problem, tzn. dwie identyczne nazwy, użytkownik zostanie zapytany, czy zastąpić plik istniejący w katalogu głównym plikiem przenoszonym. Jeśli odpowiedź będzie twierdząca, plik zostanie przeniesiony a folder usunięty. W przeciwnym wypadku, plik nie zostanie przeniesiony a katalog, będzie istniał. Program obsługuje także linki symboliczne do plików. Jeśli zostanie napotkany link do pliku, to zostaje usunięty i utworzony w katalogu docelowym. Funkcja obsługująca podane operacje jest zdefiniowana następująco:

```
function flatten {
    local catalog=$1
    local depth=$2

    if [ $depth <> 0 ] && [ $catalog != $main ]; then
        mv -i -v 'find $catalog -maxdepth 1 -type f' $main
        for i in `find "$catalog-maxdepth 1 -type l`; do
            name=$i
            name=`echo $i | awk 'BEGIN {FS="/"} {print $ NF}'`
            toLink=`readlink $i`
            ln -s $toLink $main/$name
            rm $i
        done
    fi

    for i in `find $catalog -mindepth 1 -maxdepth 1 -type d`; do
        if [ $maxDepth = -1 ] || [ $depth -lt $maxDepth ]; then
            flatten $i $((depth + 1))
            rmdir $i 2>/dev/null
        fi
    done
}
```

#### 2.3. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
flatten.sh [ścieżka do katalogu]
```

## 2.4. Przykład

Przykład użycia skryptu spłaszczenia katalogu. Argumentem skryptu jest ścieżka względna do *Katalog*.

```
kamil@kamil-Satellite-A300:~$ ls -lR ./Katalog
./Katalog:
razem 12
drwxrwxr-x 2 kamil kamil 4096 sty 17 19:34 c
drwxrwxr-x 2 kamil kamil 4096 sty 17 19:34 doc
drwxrwxr-x 2 kamil kamil 4096 sty 17 19:34 txt

./Katalog/c:
razem 0
-rw-rw-r-- 1 kamil kamil 0 wrz  2  2011 Plik2.c

./Katalog/doc:
razem 0
-rw-rw-r-- 1 kamil kamil 0 sty 17 18:50 Plik3.doc

./Katalog/txt:
razem 0
-rw-rw-r-- 1 kamil kamil 0 sie  1  2010 Plik1.txt
-rw-rw-r-- 1 kamil kamil 0 sty 17 18:50 Plik4.txt
```

(a) Przed uruchomieniem skryptu.

```
kamil@kamil-Satellite-A300:~$ bash flatten.sh ./Katalog
'/home/kamil/Katalog/txt/Plik1.txt' -> '/home/kamil/Katalog/Plik1.txt'
'/home/kamil/Katalog/txt/Plik4.txt' -> '/home/kamil/Katalog/Plik4.txt'
'/home/kamil/Katalog/doc/Plik3.doc' -> '/home/kamil/Katalog/Plik3.doc'
'/home/kamil/Katalog/c/Plik2.c' -> '/home/kamil/Katalog/Plik2.c'
```

(b) Efekt uruchomienia skryptu.

Rysunek 2.1. Spłaszczenie katalogu.

### 3. Katalog dwupoziomowy

W katalogu jednopoziomowym użytkownik musi korzystać z jednego katalogu. Powoduje to konflikty w nazywaniu plików. Problem ten rozwiązano wprowadzając dwupoziomową strukturę katalogów. Katalog główny zawiera szereg podkatalogów. W podkatalogach znajdują się pliki – bez możliwości tworzenia dalszych podkatalogów. W takiej strukturze plik jest jednoznacznie identyfikowany przez podanie nazwy podkatalogu i nazwy pliku.

#### 3.1. Katalogowanie plików wg daty modyfikacji

Skrypt pozwala na stworzenie katalogu dwupoziomowego. Wszystkie pliki znajdujące się w podanej wcześniej lokalizacji, zostaną przeniesione do odpowiednich katalogów.

##### 3.1.1. Opis skryptu

Skrypt przyjmuje jeden argument: ścieżkę do katalogu, w którym znajdują się pliki do posegregowania. Operacja katalogowania wygląda następująco – dla każdego pliku znajdującego się w katalogu podanym w argumencie wywołania skryptu, jest sprawdzany czas modyfikacji. Następnie na podstawie znalezionej daty modyfikacji, zostanie utworzony folder, do którego będą przenoszone odpowiednie pliki. Funkcja jest zdefiniowana następująco:

```
files='ls -l --time-style=long-iso |grep -e '^.*\.'|awk 'print $6, $8'
echo 'files ' | while read dates name
do
    dates='echo "$dates" | cut -d - -f -2'
    if [ ! -d "$dates" ]; then
        mkdir "$dates"
    fi
    echo Created $dates
    mv -v $name $dates/$name
done
```

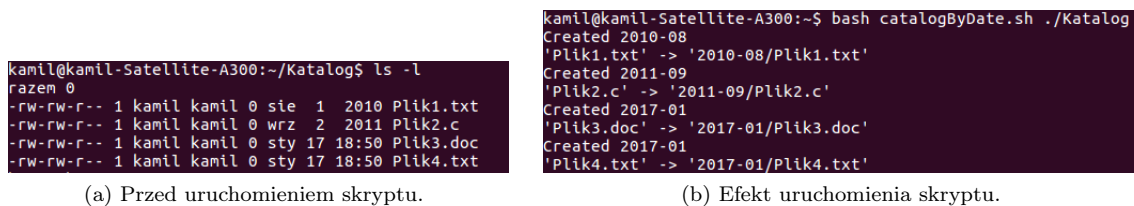
##### 3.1.2. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
catalogByDate.sh [ścieżka do katalogu]
```

##### 3.1.3. Przykład

Przykład użycia skryptu do katalogowania plików wg ostatniej daty modyfikacji pliku. Argumentem skryptu jest ścieżka względna do *Katalog*.



Rysunek 3.1. Katalogowanie plików wg daty modyfikacji.

### 3.2. Katalogowanie plików wg rozszerzenia

Skrypt pozwala na stworzenie katalogu dwupoziomowego. Wszystkie pliki znajdujące się w podanej wcześniej lokalizacji, zostaną przeniesione do odpowiednich katalogów.

#### 3.2.1. Opis skryptu

Skrypt przyjmuje jeden argument: ścieżkę do katalogu, w którym znajdują się pliki do posegregowania. Operacja katalogowania wygląda następująco – dla każdego pliku znajdującego się w katalogu podanym w argumencie wywołania skryptu, jest sprawdzane rozszerzenie. Następnie na podstawie rozszerzenia pliku zostanie utworzony folder, do którego będą przenoszone odpowiednie pliki. Funkcja jest zdefiniowana następująco:

```
exts='$(ls |grep -e '^.*\.'| sed 's/^.*\./' | sort -u)'
for ext in $exts
do
    echo Created $ext
    mkdir $ext
    mv -v *.$ext $ext/
done
```

### 3.2.2. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
catalogByExtension.sh [ścieżka do katalogu]
```

### 3.2.3. Przykład

Przykład użycia skryptu do katalogowania plików wg rozszerzenia pliku. Argumentem skryptu jest ścieżka względna do *Katalog*.

```
kamil@kamil-Satellite-A300:~/Katalog$ ls -l
razem 0
-rw-rw-r-- 1 kamil kamil 0 sie  1  2010 Plik1.txt
-rw-rw-r-- 1 kamil kamil 0 wrz  2  2011 Plik2.c
-rw-rw-r-- 1 kamil kamil 0 sty 17 18:50 Plik3.doc
-rw-rw-r-- 1 kamil kamil 0 sty 17 18:50 Plik4.txt
```

(a) Przed uruchomieniem skryptu.

```
kamil@kamil-Satellite-A300:~$ bash flatten.sh ./Katalog
'/home/kamil/Katalog/txt/Plik1.txt' -> '/home/kamil/Katalog/Plik1.txt'
'/home/kamil/Katalog/txt/Plik4.txt' -> '/home/kamil/Katalog/Plik4.txt'
'/home/kamil/Katalog/doc/Plik3.doc' -> '/home/kamil/Katalog/Plik3.doc'
'/home/kamil/Katalog/c/Plik2.c' -> '/home/kamil/Katalog/Plik2.c'
```

(b) Efekt uruchomienia skryptu.

Rysunek 3.2. Katalogowanie plików wg rozszerzenia.

## 4. Drzewiasta struktura katalogów

Pomysł grupowania plików w podkatalogi łatwo uogólnić. Katalog może zawierać pliki i podkatalogi, a cała struktura katalogów może mieć kształt drzewa, którego korzeń stanowi główny katalog. Pliki są identyfikowane poprzez podanie ścieżki w drzewie (złożonej z nazw podkatalogów) oraz nazwy pliku.

### 4.1. Struktura grafu acyklicznego

W strukturze tej dopuszczamy, aby katalogi i pliki tworzyły graf acykliczny. Oznacza to, że każdy plik i podkatalog (ale nie katalog główny) występuje w jednym lub więcej katalogach. Warunek acykliczności gwarantuje nam, że z katalogu głównego możemy dotrzeć do każdego katalogu i pliku. Przy takiej strukturze katalogów mamy nową operację: wstawienie istniejącego pliku lub podkatalogu do określonego katalogu. W wyniku wykonania takiej operacji mamy więcej niż jedno dowiązanie do danego pliku/podkatalogu. Ponadto plik/katalog ten może w różnych katalogach pojawiać się pod różnymi nazwami.

### 4.2. Zapisywanie struktury do pliku

Skrypt pozwala na wyświetlenie wykresu, na którym pokazana jest struktura katalogów i plików na dysku twardym. Obsługuje także linki symboliczne.

#### 4.2.1. Opis skryptu

Skrypt przyjmuje jeden argument: ścieżkę do katalogu, który oznacza korzeń drzewa katalogów. W programie zostały zdefiniowane dwie zmienne globalne: *olddir* – ścieżka do lokalizacji z której został wywołany skrypt, *depth* – aktualna głębokość w strukturze folderów. Operacja tworzenia pliku tekstowego ze strukturą katalogów i plików jest zdefiniowana przez funkcję *listfiles* i wygląda następująco – dla każdego podkatalogu w drzewie katalogu podanego w argumencie, obojętnie na którym poziomie, zostaną wybrane wszystkie pliki i wpisane informacje do pliku tekstowego. Jeśli jest to katalog to zostanie zwiększony licznik głębokości *depth* i wywołana funkcja rekurencyjnie *listfiles*. Funkcja *listfiles* jest zdefiniowana następująco:

```
function listfiles {
  cd $1
  for file in * do
    for ((i=0; $i < $depth; i++)) do
      if [ $i = $($depth-1) ] ; then
        echo -n '|——'»$olddir/tree.txt
      else
        echo -n '|      '»$olddir/tree.txt
      fi
    done

    if [ 'readlink $file' != '' ]; then
      echo $file -> 'readlink $file'»$olddir/tree.txt
    elif [ -d "$file" ]; then
      echo [ $file ]»$olddir/tree.txt
    else
      echo $file»$olddir/tree.txt
    fi

    if [ -d "$file" ] && [ 'readlink $file' = '' ]; then
      depth=$((depth+1));
      listfiles "$file";
      cd ..;
    fi
  done
  depth=$((depth-1));
}
```

Wszystkie informacje o plikach są zapisywane do pliku tekstowego o nazwie – *tree.txt*.



### 4.2.2. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
treeToFile.sh [ścieżka do katalogu]
```

### 4.2.3. Przykład

Przykład użycia skryptu, gdzie argumentem jest ścieżka względna do *Katalog1*.

```
kamil@kamil-Satellite-A300:~/Katalog1$ ls -lR
.:
razem 8
drwxrwxr-x 2 kamil kamil 4096 sty 17 18:32 Katalog2
drwxrwxr-x 2 kamil kamil 4096 sty 17 18:32 Katalog3
-rw-rw-r-- 1 kamil kamil  0 sty 17 18:32 Plik1

./Katalog2:
razem 0

./Katalog3:
razem 0
lrwxrwxrwx 1 kamil kamil 29 sty 17 18:32 Link1 -> /home/kamil/Katalog1/Katalog2
```

(a) Przykład struktury grafu acyklicznego.

```
kamil@kamil-Satellite-A300:~$ bash treeToFile.sh ./Katalog1
/home/kamil/Katalog1
----- [ Katalog2 ]
|----- *
|----- [ Katalog3 ]
|----- Link1 -> /home/kamil/Katalog1/Katalog2
|----- Plik1
```

(b) Efekt uruchomienia skryptu.

Rysunek 4.1. Wyświetlanie zawartości podanego katalogu i wszystkich jego podkatalogów w formie drzewa.

## 4.3. Tworzenie struktury z pliku

Skrypt pozwala na utworzenie struktury plików i katalogów, na podstawie wcześniej utworzonego wykresu.

### 4.3.1. Opis skryptu

Skrypt przyjmuje jeden argument: ścieżkę do pliku, z którego będzie tworzona struktura. W programie zostały zdefiniowane trzy funkcje: *createFolder* – tworzy katalog o nazwie podaną w argumencie funkcji, *createFile* – tworzy plik o nazwie podaną w argumencie oraz *entry* – przechodzi do katalogu o nazwie podanej przez argument funkcji. Wszystkie trzy podane funkcje są odporne na błędy. Operacja tworzenia struktury katalogów i plików wygląda następująco – za pomocą polecenia *awk* skrypt interpretuje pojedynczo ciąg linii z pliku, gdzie po każdej linii przekazuje informacje co należy wykonać. Takimi informacjami są:

- czy należy wejść do katalogu wcześniej utworzonego;
- czy należy wyjść do katalogu nadrzędnego;
- czy wczytany element jest plikiem;
- czy wczytany element jest katalogiem;
- czy wczytany element jest linkiem, jeśli tak to zapisuje ścieżkę na co wskazuje;

### 4.3.2. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
treeFromFile.sh [plik tekstowy]
```

Gdzie ścieżka do pliku tekstowego powinna być względna od bieżącego położenia.

### 4.3.3. Przykład

Przykład użycia skryptu, gdzie argumentem jest ścieżka względna do pliku tekstowego *tree.txt*.

```
kamil@kamil-Satellite-A300:~$ bash treeFromFile.sh ./tree.txt
/home/kamil/Katalog1
|----- [ Katalog2 ]
|         |----- *
|         |----- [ Katalog3 ]
|         |         |----- Link1 -> /home/kamil/Katalog1/Katalog2
|         |         |----- Plik1
```

(a) Uruchomienie skryptu.

```
kamil@kamil-Satellite-A300:~/Katalog1$ ls -lR
.:
razem 8
drwxrwxr-x 2 kamil kamil 4096 sty 17 18:32 Katalog2
drwxrwxr-x 2 kamil kamil 4096 sty 17 18:32 Katalog3
-rw-rw-r-- 1 kamil kamil  0 sty 17 18:32 Plik1

./Katalog2:
razem 0

./Katalog3:
razem 0
lrwxrwxrwx 1 kamil kamil 29 sty 17 18:32 Link1 -> /home/kamil/Katalog1/Katalog2
```

(b) Efekt uruchomienia skryptu.

Rysunek 4.2. Tworzenie struktury katalogów i plików na podstawie zawartości pliku tekstowego.

## 5. Proste operacje na plikach

Poniżej znajdują się skrypty przydatne dla każdego użytkownika systemu Linux, operujące na plikach.

### 5.1. Synchronizacja danych

Skrypt pozwala na synchronizację dwóch lokalizacji, uwzględniając:

- kopiowanie podkatalogów;
- pokazuje podczas wykonywania, które pliki są synchronizowane;
- zachowanie czasów pliku źródłowego;
- pomijanie plików starszych, które są nowsze w lokalizacji odbierającej dane;

Funkcja synchronizująca katalogi jest zdefiniowana następująco:

```
chmod -R 777 $first
chmod -R 777 $second
rsync -v -r -u -t $first $second
sync
notify-send "Updated catalog 1"
rsync -v -r -u -t $second $first
sync
notify-send "Updated catalog 2"
```

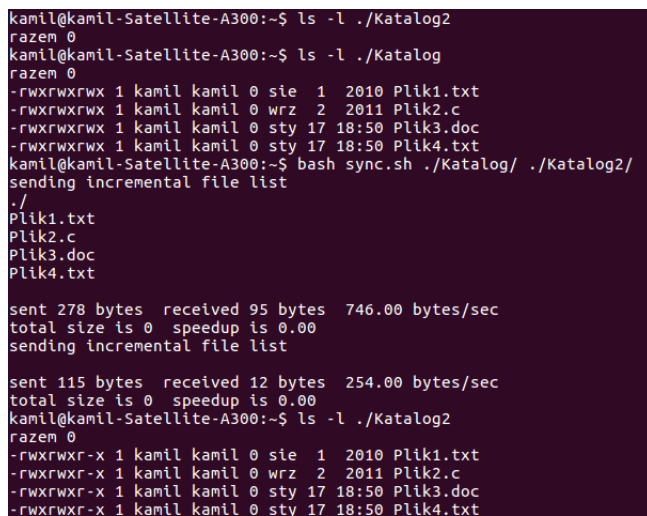
#### 5.1.1. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
sync.sh [pierwsza lokalizacja] [druga lokalizacja]
```

#### 5.1.2. Przykład

Przykład użycia skryptu do synchronizacji dwóch lokalizacji. Skrypt został uruchomiony z dwoma argumentami – dwie różne lokalizacje.



```
kamil@kamil-Satellite-A300:~$ ls -l ./Katalog2
razem 0
kamil@kamil-Satellite-A300:~$ ls -l ./Katalog
razem 0
-rwxrwxrwx 1 kamil kamil 0 sie  1  2010 Plik1.txt
-rwxrwxrwx 1 kamil kamil 0 wrz  2  2011 Plik2.c
-rwxrwxrwx 1 kamil kamil 0 sty 17 18:50 Plik3.doc
-rwxrwxrwx 1 kamil kamil 0 sty 17 18:50 Plik4.txt
kamil@kamil-Satellite-A300:~$ bash sync.sh ./Katalog/ ./Katalog2/
sending incremental file list
./
Plik1.txt
Plik2.c
Plik3.doc
Plik4.txt

sent 278 bytes  received 95 bytes  746.00 bytes/sec
total size is 0  speedup is 0.00
sending incremental file list

sent 115 bytes  received 12 bytes  254.00 bytes/sec
total size is 0  speedup is 0.00
kamil@kamil-Satellite-A300:~$ ls -l ./Katalog2
razem 0
-rwxrwxr-x 1 kamil kamil 0 sie  1  2010 Plik1.txt
-rwxrwxr-x 1 kamil kamil 0 wrz  2  2011 Plik2.c
-rwxrwxr-x 1 kamil kamil 0 sty 17 18:50 Plik3.doc
-rwxrwxr-x 1 kamil kamil 0 sty 17 18:50 Plik4.txt
```

Rysunek 5.1. Synchronizacja dwóch lokalizacji.

Możemy zauważyć, że na początku w *Katalog2* nie znajdowały się pliki. Dopiero po uruchomieniu skryptu, w *Katalog2* pojawiły się te same pliki co w lokalizacji *Katalog*.

### 5.2. Monitorowanie miejsca na dysku

Skrypt pozwala na monitorowanie miejsca dostępnego na dysku. Przyjmuje jeden argument, liczbę naturalną, określającą maksymalny procent zajmowania danych na dysku. Jeśli ilość zajmowanych danych na dysku przekroczy dany próg, wtedy użytkownik zostanie poinformowany specjalnym komunikatem. Funkcja odpowiedzialna za monitorowanie zdefiniowana jest następująco:

```
path='^/dev'
df='df -h | grep $path | awk 'print $5, $6' | sed 's/%/"/'
echo '$df' | while read percent fs
do
    if [ $percent -gt $acceptable ]; then
        notify-send "It is used $percent % of the directory $fs"
    fi
done
```

### 5.2.1. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
monitor.sh [dopuszczalny próg]
```

### 5.3. Usuwanie linków

Skrypt wyszukuje w zadanym katalogu linki, które na nic nie wskazują. Po znalezieniu zostaną usunięte wraz z krótką informacją o linku. Funkcja odpowiedzialna za usuwanie linków:

```
for i in `find $catalog -type l`; do
    if [ ! -f `readlink $i` -a ! -d `readlink $i` ]; then
        echo -n "$i -> `readlink $i`"
        rm -f $i 2>/dev/null
        if [ -L "$i" ]; then
            echo -n ' not deleted'
        else
            echo -n ' deleted'
        fi
        echo
    fi
done
```

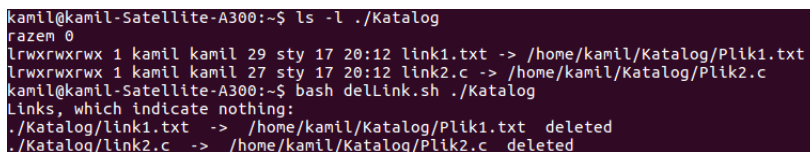
#### 5.3.1. Sposób uruchamiania

Aby prawidłowo uruchomić podany skrypt należy wpisać w terminal:

```
delLink.sh [ścieżka do katalogu]
```

#### 5.3.2. Przykład

Przykład użycia skryptu, w celu usunięcia wszystkich nieaktywnych linków w lokalizacji podanej jako argument skryptu.



```
kamil@kamil-Satellite-A300:~$ ls -l ./Katalog
razem 0
lrwxrwxrwx 1 kamil kamil 29 sty 17 20:12 link1.txt -> /home/kamil/Katalog/Plik1.txt
lrwxrwxrwx 1 kamil kamil 27 sty 17 20:12 link2.c -> /home/kamil/Katalog/Plik2.c
kamil@kamil-Satellite-A300:~$ bash delLink.sh ./Katalog
Links, which indicate nothing:
./Katalog/link1.txt -> /home/kamil/Katalog/Plik1.txt deleted
./Katalog/link2.c -> /home/kamil/Katalog/Plik2.c deleted
```

Rysunek 5.2. Usuwanie linków.

## 6. Podsumowanie

Aby wykorzystać efektywniej skrypty, można skorzystać z *crontab* – ważny demon systemowy, którego zadaniem jest uruchamianie programów cyklicznie lub o określonej porze. Aby skrypt był wywoływany co 5 min należy wykonać poniżej instrukcje:

- Wstawić skrypt do /usr/bin;
- Nadać uprawnienia pliku do wykonywania;
- Wprowadzić wpis do tablicy crontab(polecenie crontab -e):

```
* /5 * * * * monitor.sh 10
```

## Literatura

- [1] System plików, edu.pjwstk.edu.pl, <http://edu.pjwstk.edu.pl/wyklady/sop/scb/wyklad9/index.html>