Linux
Pliki i katalogi (podstawowe polecenia)

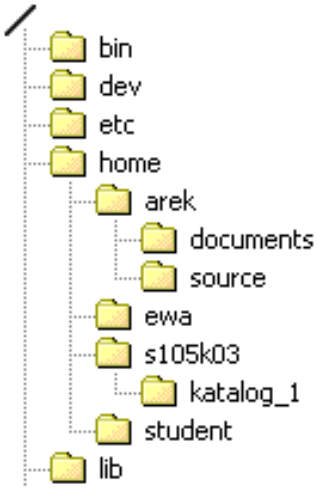
Opracował: Arkadiusz Curulak
WSIiE TWP w Olsztynie

Data aktualizacji : **17-06-2002**
Pierwsza edycja : 26-04-2002

Spis treści

Linux: Pliki i katalogi (podstawowe polecenia)

W systemie Linux pliki przechowywane są w hierarchicznie połączonych ze sobą katalogach. Struktura plików (i katalogów) nazywana jest strukturą drzewiastą. Katalogiem głównym, od którego rozpoczyna się struktura na dysku jest /.



Rys.1. Przykładowa struktura plików.

W katalogu *home* zlokalizowane są katalogi domowe (macierzyste) użytkowników. W tych katalogach użytkownicy mogą tworzyć własne katalogi i przechowywać swoje pliki. Po zalogowaniu się do systemu, katalog macierzysty staje się automatycznie katalogiem roboczym (bieżącym).

Przykładowo: katalogiem domowym użytkownika *arek* jest katalog nazywający się tak samo jak jego login (nazwa użytkownika) *arek*. Pełna ścieżka dostępu do tego katalogu to */home/arek* .

Zarządzanie katalogami

Zanim przedstawię podstawowe polecenia wykorzystywane do zarządzania katalogami w systemie Linux, zwrócę uwagę na bardzo przydatny parametr, na który reaguje większość poleceń. Mowa o `--help`. Wywołanie polecenia z tym parametrem wyświetla krótki opis tego polecenia (do czego służy, składnię, parametry).

Przykładowo: nie pamiętamy składni polecenia *rmdir*. Zatem ...

```
[arek@student ~] $ rmdir --help

Usage: rmdir [OPTION]... DIRECTORY...
Remove the DIRECTORY(ies), if they are empty.

--ignore-fail-on-non-empty
                ignore each failure that is solely because
                a directory is non-empty
-p, --parents   remove DIRECTORY, then try to remove each directory
                component of that path name.  E.g., `rmdir -p a/b/c'
                is similar to `rmdir a/b/c a/b a'.
-v, --verbose   output a diagnostic for every directory processed
--help          display this help and exit
--version       output version information and exit
```

A teraz opis podstawowych poleceń.

- **pwd** - podanie bieżącej lokalizacji (bieżącego katalogu).
  Składnia: `pwd`

```
[arek@student ~] $ pwd
/home/arek
```

`[arek@student ~] $` - jest znakiem zachęty informującym o gotowości systemu do przyjmowania poleceń. Znak zachęty może wyglądać różnie, w zależności od jego definicji. W naszym przypadku wygląda tak jak widać i niesie ze sobą informacje: `arek@student` - user *arek* na serwerze *student*; `~` - nazwa bieżącego katalogu (`~` oznacza katalog domowy, czyli `/home/arek`); `$` - dodatkowy znak kończący znak zachęty.

- **mkdir** - utworzenie nowego katalogu.
  Składnia: `mkdir nazwa_katalogu`

Utworzenie w katalogu bieżącym nowego katalogu o nazwie *raporty*.
```
[arek@student ~] $ mkdir raporty
```

Utworzenie w katalogu */home/arek/documents* nowego katalogu o nazwie *listy*.
```
[arek@student ~] $ mkdir /home/arek/documents/listy
```

- **rmdir** - usunięcie pustego katalogu.
  Składnia: `rmdir nazwa_katalogu`

Usunięcie z katalogu bieżącego katalogu o nazwie *kosz*.
```
[arek@student ~] $ rmdir kosz
```

Usunięcie z katalogu */home/arek/source* katalogu o nazwie *test*.
```
[arek@student ~] $ rmdir /home/arek/source/test
```

Uwaga! Powyższe polecenia *rmdir* usuną katalogi pod warunkiem, że będą ona puste. W przeciwnym razie zobaczymy komunikat o błędzie:
```
rmdir: 'kosz': Directory not empty
```
(lub komunikat podobny).

- **cd** - zmiana bieżącego katalogu.
  Składnia: `cd nazwa_katalogu`

Przejście do katalogu *documents* (patrz rys.1).
```
[arek@student ~] $ cd /home/arek/documents
```

Wiemy, że katalog *documents* znajduje się w naszym katalogu domowym, możemy zatem wydać krótsze polecenie:
```
[arek@student ~] $ cd ~/documents
```

Jeżeli bieżącym katalogiem jest właśnie nasz katalog domowy */home/arek*, co możemy sprawdzić poleceniem *pwd* (o ile znak ~ w znaku zachęty nie przemawia do nas), to możemy wydać jeszcze krótsze polecenie:

```
[arek@student ~] $ cd documents
```

Przejście do katalogu głównego naszego systemu plików.
```
[arek@student ~] $ cd /
```

Powrót do katalogu nadrzędnego w stosunku do katalogu bieżącego.
```
[arek@student ~] $ cd ..
```

Przykładowo: po wydaniu polecenia:
```
[arek@student ~] $ cd /home/arek/documents
```
"lądujemy" w katalogu *documents*. Jeżeli chcemy powrócić do katalogu nadrzędnego (w naszym przypadku będzie to /home/arek), wydajemy polecenie:
```
[arek@student ~] $ cd ..
```

Szybki powrót do katalogu domowego:
```
[arek@student ~] $ cd ~
```

Możemy również użyć wersji skróconej (samo polecenie *cd*):
```
[arek@student ~] $ cd
```

Przykład:
Bieżącym katalogiem jest */home/arek/documents* (patrz rys.1). Chcę wejść do katalogu */home/arek/source*. Wydaję zatem polecenie:
```
[arek@student ~] $ cd ../source
```
Najpierw wchodzę do katalogu nadrzędnego (*/home/arek*), a potem do właściwego katalogu *source*.

Pamiętamy, że poprawność użycia polecenia *cd* możemy sprawdzić poleceniem *pwd*, które powie nam, w jakim katalogu obecnie jesteśmy.

- **`ls`** - wylistowanie zawartości katalogu.
  Składnia: `ls [nazwa_katalogu]`

Wylistowanie zawartości bieżącego katalogu (nie wymaga podawania nazwy katalogu).
```
[arek@student ~] $ ls
```

W odpowiedzi zobaczymy listę plików i katalogów znajdujących się w bieżącym katalogu (wyświetloną alfabetycznie):
```
documents  kto  licznik.c  podanie  source  wyniki.dat
```

O ile system nie jest inaczej skonfigurowany, wyniki polecenia *ls* będą wyświetlane w postaci jak wyżej. Aby rozróżniać katalogi i pliki, trzeba użyć parametru `-F`.

```
[arek@student ~] $ ls -F
documents/  kto  licznik.c  podanie  source/  wyniki.dat
```

Teraz już wszystko jasne (katalogami są *documents* i *source*).

Jeżeli chcemy przeglądać zawartość katalogu, który nie jest katalogiem bieżącym (roboczym), musimy po poleceniu *ls* podać ścieżkę dostępu do tego katalogu.

Przykładowo: wyświetlmy zawartość katalogu *katalog_1* (patrz rys.1). Musimy zatem wydać polecenie:

```
[arek@student ~] $ ls /home/s105k03/katalog_1
```

Jeżeli chcemy uzyskać więcej informacji nt. wylistowanych plików i katalogów, polecenie *ls* wywołujemy z parametrem `-l`.

```
[arek@student ~] $ ls -l /home/arek
```

W odpowiedzi zobaczymy:

```
drwxr-xr-x  3  arek  users   4096 Feb 21 14:45 documents
-rwxr-xr-x  1  arek  users    716 Apr 22 18:12 kto
-rw-r--r--  1  arek  users   2874 Mar 26 10:46 licznik.c
-rw-r--r--  1  arek  users   1170 Apr 25 09:31 podanie
drwxr-xr-x  2  arek  users   4096 Mar 11 07:13 source
-rw-r--r--  1  arek  users    534 Apr 27 02:02 wyniki.dat
lrwxrwxrwx  1  arek  users     30 May 17 13:01 go -> /usr/local/sbin/go.sh
```

Pierwszy zestaw znaków oznacza prawa dostępu do pliku lub katalogu, przy czym po pierwszym znaku możemy domyślić się, z jakim obiektem mamy do czynienia (`d` (directory) katalog, `-` (minus) plik, `l` (link) dowiązanie symboliczne). W następnej kolumnie występuje liczba dowiązań do obiektu. Dalej mamy nazwę użytkownika/właściciela obiektu i grupę, która ma prawa dostępu do obiektu (prawa te wyspecyfikowane są w części praw dla grupy). Po nazwie grupy występuje rozmiar obiektu w bajtach, a następnie data i czas ostatniej modyfikacji. Na końcu każdego wiersza wypisana jest nazwa obiektu (pliku lub katalogu).

Jeżeli lista plików i katalogów jest tak długa, że nie mieści się jednorazowo na ekranie, możesz zażądać stronicowania wyświetlanej informacji. Do tego celu przygotowane zostały polecenia *more* i *less*.

Przykładowo: wyświetlmy zawartość katalogu */etc* z pełnymi informacjami o plikach i katalogach:

```
[arek@student ~] $ ls -l /etc | more
```

Teraz po zapełnieniu ekranu informacjami, polecenie *more* zatrzyma się w oczekiwaniu na wydanie polecenia wyświetlenia kolejnej strony. Naciśnięcie *spacji* wyświetli kolejną stronę, a *Enter* kolejny wiersz. O tym, że jest jeszcze coś do wyświetlenia mówi napis `--More--` wypisywany na końcu każdej strony.

Zarządzanie plikami

Omawianie podstawowych poleceń służących do zarządzania plikami, zaczniemy od polecenia tworzenia pliku.

- tworzenie nowego pliku.

Sposobów na utworzenie pliku mamy kilka. Zaczniemy od poleceń tworzących puste pliki (bez wprowadzania do nich danych).

```
[arek@student ~] $ touch nazwa_pliku
```

Wprawdzie polecenie *touch* ma szersze zastosowanie, jednak świetnie nadaje się również do tworzenia pustego pliku.

```
[arek@student ~] $ > nazwa_pliku
```

Powyżej widzimy szybki sposób na utworzenie pustego pliku z wykorzystaniem techniki przekierowywania strumieni, o której jeszcze sobie powiemy.

Teraz kilka przykładów:

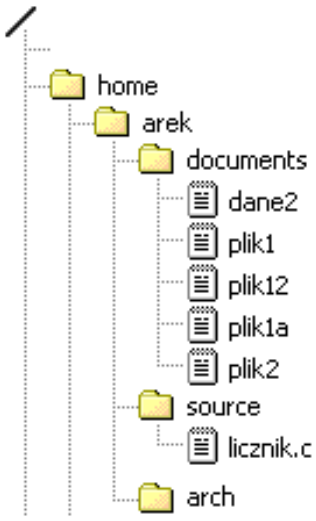
```
[arek@student ~] $ touch wyniki.dat
[arek@student ~] $ touch /home/arek/documents/raport
[arek@student ~] $ > list.txt
[arek@student ~] $ > ~/source/licznik.c
```

Ostatnim z omawianych tu sposobów, będzie utworzenie nowego pliku tekstowego z jednoczesnym wpisaniem do niego danych tekstowych. Wykorzystamy do tego celu polecenie *cat* oraz przekierowanie strumienia.

```
[arek@student ~] $ cat > nazwa_pliku
pierwszy wiersz tekstu
drugi wiersz tekstu
trzeci wiersz tekstu
^D
```

Jak widać, po wydaniu polecenia, kursor ustawia się na początku następnego wiersza w oczekiwaniu na edycję. Po wprowadzeniu tekstu musimy ustawić znak końca pliku (`^D`). Robimy to przez naciśnięcie kombinacji klawiszy CTRL+D.

Podczas omawiania kolejnych poleceń posłużymy się następującą strukturą katalogów i plików (w zasadzie wycinkiem zawierającym nasz katalog domowy):



Rys.2. Przykładowy fragment struktury plików i katalogów.

- **`cp`** - kopiowanie plików.
  Składnia: `cp plik_źródłowy plik_docelowy`

Skopiowanie pliku *dane2* do katalogu *arch*.
```
[arek@student ~] $ cp ~/documents/dane2 ~/arch
```

Jako, że bieżącym katalogiem jest */home/arek* oraz poruszamy się właśnie w tym katalogu (naszym katalogu domowym (patrz rys.2)), powyższe polecenie możemy uprościć:
```
[arek@student ~] $ cp documents/dane2 arch
```

Skopiowanie wszystkich plików z katalogu */home/arek/documents* do katalogu */home/arek*. Oto możliwe postaci polecenia:

```
[arek@student ~] $ cp /home/arek/documents/* /home/arek
[arek@student ~] $ cp ~/documents/* ~
[arek@student ~] $ cp documents/* .
```

W trzecim przykładzie kropka (`.`) oznacza katalog bieżący, czyli wszystkie pliki z katalogu *documents* zostaną skopiowane do katalogu bieżącego.

Wszystkie powyższe polecenia odnosiły się wyłącznie do plików. Jeżeli zatem w katalogu *documents* znajdowałyby się jakieś podkatalogi z plikami, byłyby one podczas kopiowania zignorowane. Aby jednak skopiować zawartość całego katalogu, wraz z podkatalogami i ich zawartością, musimy użyć parametru `-r`.

Przykładowo:
```
[arek@student ~] $ cp -r ~/documents/* ~/arch
```

- **`mv`** - zmiana nazwy pliku (lub katalogu) lub przenoszenie plików (i katalogów).
  Składnia: `mv oryginalna_nazwa_pliku nowa_nazwa_pliku`
  Składnia: `mv plik_źródłowy katalog_docelowy`

Polecenie *mv* ma dwojakie znaczenie w zależności od sposobu jego użycia.

Zmiana nazwy pliku *dane2* na *wynik.txt*.
```
[arek@student ~] $ mv ~/documents/dane2 ~/documents/wynik.txt
```

Z polecenia *mv* trzeba korzystać z uwagą, aby nie użyć go w znaczeniu innym niż założymy. Jeżeli bowiem powyższe polecenie skrócimy do takiego:

```
[arek@student ~] $ mv ~/documents/dane2 wynik.txt
```

przeniesiemy plik *dane2* do katalogu bieżącego z nową nazwą *wynik.txt* (w katalogu *documents* pliku *dane2* już nie będzie).

Przenieśmy teraz wszystko z katalogu */home/arek/source* do katalogu */home/arek/arch*.

```
[arek@student ~] $ mv /home/arek/source/* /home/arek/arch
```

- **`rm`** - kasowanie pliku.
  Składnia: `rm nazwa_pliku`

Usunięcie pliku *licznik.c* (patrz rys.2).
```
[arek@student ~] $ rm /home/arek/source/licznik.c
```

Usunięcie jednocześnie plików *plik1* oraz *plik2* (patrz rys.2).
```
[arek@student ~] $ rm documents/plik1 documents/plik2
```

Usunięcie wszystkich plików z */home/arek/documents* (patrz rys.2).
```
[arek@student ~] $ rm documents/*
```

Usunięcie wszystkiego (plików i ewentualnych katalogów) z */home/arek/documents*.
```
[arek@student ~] $ rm –r documents/*
```

Jak widać, aby usunąć całą zawartość jakiegoś katalogu (łącznie z podkatalogami), polecenie *rm* musimy wywołać z parametrem `–r` (lub `–R`).

- **`cat`** - przeglądanie zawartości pliku tekstowego.
  Składnia: `cat nazwa_pliku`

Wyświetlenie zawartości pliku *licznik.c* (patrz rys.2).
```
[arek@student ~] $ cat /home/arek/source/licznik.c
[arek@student ~] $ cat source/licznik.c
```

Jeżeli treść pliku nie mieści się w całości na ekranie, możemy użyć znanego już nam stronicowania za pomocą polecenia *more*.
```
[arek@student ~] $ cat source/licznik.c | more
```

- **`ln`** - utworzenie dowiązania (linku).
  Składnia: `ln oryginalna_nazwa_obiektu nazwa_dowiązania`

Dzięki dowiązaniom, do danego pliku lub katalogu, możemy odwoływać się za pomocą kilku nazw i z różnych miejsc struktury katalogowej.
Załóżmy, że bardzo często będziemy odwoływali się do pliku *licznik.c* (rys.2), bo właśnie nad nim pracujemy. Wygodniej będzie nam z pewnością używać krótkiej nazwy, np. *licz.c*, niż najkrótszej możliwej nazwy pozwalającej odwołać się wprost do oryginału *licznik.c*, a mianowicie *source/licznik.c*. Tworzymy zatem sobie skrót za pomocą polecenia *ln*.

```
[arek@student ~] $ ln source/licznik.c licz.c
```

Teraz chcąc na przykład wyświetlić zawartość pliku *source/licznik.c* wystarczy, że podamy polecenie:
```
[arek@student ~] $ cat licz.c
```
ponieważ w katalogu `~` (*/home/arek*, który jest naszym katalogiem roboczym) znajduje się teraz plik *licz.c*, który w rzeczywistości jest dowiązaniem do pliku *source/licznik.c*, które mówi, na którym pliku w rzeczywistości należy wykonać operacje.

Mówiąc o dowiązaniach, należy wspomnieć o istnieniu dwóch rodzajów dowiązań: *sztywnych* i *symbolicznych*. Różnica między nimi polega na tym, że dowiązania sztywne działają jedynie w obrębie tego samego systemu plików. Jeżeli wiec będziemy próbowali utworzyć dowiązanie sztywne do pliku znajdującego się na innej partycji bieżącego dysku lub w ogóle na innym dysku, możemy spotkać się z komunikatem:

```
ln: creating hard link `test.txt' to `/mnt/hda3': Invalid cross-device link
```

Tworząc w tej samej sytuacji dowiązanie symboliczne, wszystko wykona się poprawnie. Zanim powiemy sobie jak utworzyć dowiązanie symboliczne, warto zwrócić uwagę na jeszcze inne cechy dowiązań sztywnych. Jeżeli usuniemy plik, do którego istnieje dowiązanie sztywne, to w chwili usuwania tego pliku następuje zastąpienie istniejącego dowiązania treścią tego pliku – dowiązanie staje się plikiem. Jeśli więc chcemy całkowicie usunąć plik z dysku, musimy usunąć również wszystkie istniejące dowiązania sztywne. Dowiązań symbolicznych własność ta nie dotyczy. Dowiązania sztywne możemy tworzyć tylko do istniejących obiektów. Dowiązania symboliczne możemy z kolei również tworzyć dla obiektów, które nie istnieją (na razie pliku nie ma, ale za chwilę już będzie).

Dowiązania symboliczne tworzymy dodając parametr `-s`.
Przykładowo: utworzenie dowiązania symbolicznego do pliku *source/licznik.c* :

```
[arek@student ~] $ ln -s source/licznik.c licz.c
```

Dowiązania symboliczne możemy również tworzyć do katalogów. Utwórzmy zatem w */home/arek* dowiązanie do katalogu */usr/local/sbin*.

```
[arek@student ~] $ ln -s /usr/local/sbin katalog_sbin
```

Mówiąc o plikach nie sposób nie wspomnieć również o innych bardzo przydatnych poleceniach, choćby o *grep*, *sort* czy *find*. Ale o tym już w innym opracowaniu.