helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

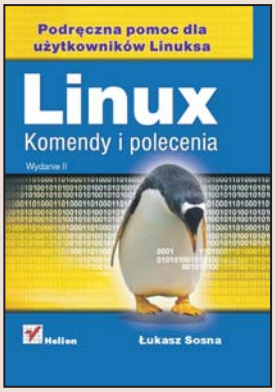
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Linux. Komendy i polecenia. Wydanie II



Autor: Łukasz Sosna
ISBN: 83-246-0636-X
Format: B5, stron: 128

Dołącz do grona fanów Linuksa

Użytkownicy Windows spotykający się po raz pierwszy z systemem Linux mogą być nieco przerażeni. Mimo graficznych narzędzi proces instalacji Linuksa jest zdecydowanie bardziej skomplikowany. Konfiguracja systemu obejmuje znacznie więcej elementów. Sposób korzystania z wielu funkcji systemu różni się zdecydowanie od tego, do czego przyzwyczaił nas Windows, a konieczność częstego stosowania konsoli tekstowej wydaje się ogromnym problemem.

Książka „Linux. Komendy i polecenia. Wydanie II" to kolejna edycja doskonałej podręcznej ściągi dla użytkowników Linuksa. Znajdziesz w niej informacje o zastosowaniu i składni poleceń systemowych. Nauczysz się korzystać z konsoli tekstowej, poznasz polecenia pozwalające na zarządzanie systemem plików, administrowanie systemem i zarządzanie kontami użytkowników, a także dowiesz się, jakich parametrów wymagają poszczególne komendy. Szybko znajdziesz wszystkie wiadomości niezbędne do sprawnego korzystania z Linuksa.

- Logowanie do systemu
- Wyłączanie i restartowanie komputera
- Struktura katalogów
- Zarządzanie systemem plików
- Administrowanie systemem
- Tworzenie skryptów powłoki
- Administrowanie kontami użytkowników

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wprowadzenie do systemu Linux 9
Czym jest Linux 9
Dostępne dystrybucje — jak wybrać odpowiednią dla siebie 10
Instalacja systemu 11

1. Korzystanie z komputera działającego pod kontrolą systemu Linux 16
Środowisko pracy 16
Logowanie się do systemu 18
Bezpieczne wyłączanie i restart komputera 20
Użytkownicy systemu Linux 20
Co znajduje się w poszczególnych katalogach systemu 21
Dyski i partycje w systemie 23
Pomoc na stronach MAN 24

2. Zarządzanie zasobami komputera 25
Pliki i katalogi w systemie 25
Wyświetlanie zawartości katalogu 27
Przechodzenie pomiędzy katalogami 36
Tworzenie katalogów 37
Usuwanie katalogów 39
Tworzenie plików 40
Usuwanie plików 40
Wyświetlanie zawartości pliku 41

Zmiana dat modyfikacji plików i dostępu do nich 42
Kopiowanie plików i katalogów 45
Przenoszenie plików i katalogów oraz zmiana ich nazwy 48
Nadawanie praw dostępu do plików i katalogów 50
Zmiana hasła 55
Zmiana powłoki 56
Uzyskiwanie informacji o typie pliku 56
Zmiana właściciela i grupy pliku 57
Wyszukiwanie plików i katalogów 59
Wypisywanie liczby bajtów, słów i linii 65
Porównywanie plików lub zakresów bajtów 66
Ustalanie zajętego i wolnego miejsca na partycjach 67
Ustalanie, ile miejsca zajmuje plik lub katalog 68
Polecenia more i less 70
Montowanie i odmontowywanie systemów plików 71
Obecna ścieżka, pod którą pracujemy 73
Przełączanie się na konto innego użytkownika 73
Uzyskiwanie informacji o sprzęcie 74
Przeglądanie kalendarza 78
Aktualizacja daty i czasu 80
Kontrolowanie wysyłania wiadomości 85
Wysyłanie wiadomości do innego użytkownika 85
Wysyłanie wiadomości z pliku tekstowego 86
Wysyłanie komunikatów do wszystkich sieci z pliku tekstowego 86
Pokazywanie ostatnio zalogowanych użytkowników 87
Sprawdzanie, kto jest aktualnie zalogowany na naszym komputerze 89
Sprawdzanie, kto jest zalogowany do systemu 90
Sprawdzanie swojej nazwy użytkownika 90
Pokazywanie lub ustawianie nazwy hosta systemowego 90
Wyświetlanie i ustalanie parametrów interfejsu sieciowego 92
Wyszukiwanie nazwy lub adresu IP zdalnego komputera 93

Sprawdzanie, czy dana domena jest już zarejestrowana 94
Polecenie sprawdzające dostępność hosta 94
Podawanie czasu, jaki upłynął od uruchomienia systemu 95

3. Administrowanie systemem ... 96
Poziom uruchomienia systemu 96
Demony usług 97
Użytkownicy 99
Grupy 101
Szukanie łańcuchów w bazie whatis 102

4. Tworzenie skryptów powłoki ... 103
Zmienne 106
Wypisywanie tekstu na ekranie użytkownika 107
Wartości logiczne 110
Polecenie test 111
Instrukcja if 116
Instrukcja case 117
Pętla while 118
Pętla until 119
Pętla for 119
Break 119
Continue 120
Argumenty pobierane z wiersza powłoki 121

Skorowidz ... 123

Rozdział 2. Zarządzanie zasobami komputera

Pierwsze, co powinieneś poznać, to zarządzanie zasobami Twojego komputera. Musisz nauczyć się tworzyć i usuwać elementy na dysku, zmieniać ich atrybuty oraz wykonywać wiele innych czynności, które pomogą Ci w poruszaniu się po strukturze katalogów i pozwolą w łatwy sposób dotrzeć do interesujących Cię plików.

Pliki i katalogi w systemie

W Linuksie, inaczej niż choćby w takich systemach operacyjnych jak systemy z rodziny Microsoft Windows, nie jest wymagane stosowanie w nazwach plików specjalnych rozszerzeń, określających m.in. to, jaki program powinien zostać użyty do otwarcia pliku. Zawartość pliku i program, jaki należy zastosować do jego otwarcia, są w Linuksie ustalane na podstawie nagłówka `MIME` pliku. Mimo to, jeżeli z jakiegokolwiek powodu odczuwasz potrzebę nadawania plikom rozszerzeń, możesz to robić — choć, jak wspomniałem, rozszerzenia te nie są konieczne, ich stosowanie nie jest także niewskazane.

W nazwach plików i katalogów możesz stosować dowolne znaki alfanumeryczne (litery i cyfry), a oprócz tego znak kropki (`.`), myślnika (`-`) i podkreślenia (`_`). Z wszystkimi innymi znakami postępuj ostrożnie; zazwyczaj są one zarezerwowane dla specjalnych funkcji systemu.

W nazwach plików i katalogów możesz używać także spacji i nie będzie to powodować większych problemów, będzie jednak po prostu niewygodne. Komendy, w których będziesz odwoływać się do plików lub katalogów zawierających spacje, będą

po prostu dłuższe i łatwiej będzie popełnić błąd podczas wpisywania nazwy pliku.

Trzeba także pamiętać o tym, że znak kropki nie powinien być znakiem rozpoczynającym nazwę pliku czy katalogu. Napisałem wprawdzie nieco wcześniej, iż jego stosowanie jest dozwolone, trzeba jednak pamiętać o tym niezwykle ważnym zastrzeżeniu: nazwy plików i katalogów nie powinny *rozpoczynać się* od znaku kropki, w każdym innym jednak miejscu może on wystąpić. Przyczyną tego ograniczenia jest to, że w Linuksie znak kropki na początku nazwy pliku jest zarezerwowany dla ukrytych plików i katalogów — na przykład plik o nazwie *.ukryty_plik* będzie plikiem ukrytym.

Niezwykle istotna jest także wielkość stosowanych przez nas liter. W omawianym systemie wielkie i małe litery są rozpoznawane jako osobne znaki. Jeśli więc tworzysz katalog lub plik, zapamiętaj, czy jego nazwę wpisałeś wielką, czy małą literą — będzie Ci to potrzebne, kiedy będziesz chciał się do niego w przyszłości odwołać.

Wyświetlany przez system znak `$` (jeśli korzystasz z konta zwykłego użytkownika) lub # (jeżeli pracujesz jako administrator systemu) jest znakiem zachęty. Oto przykład jego użycia:

```
[lukasz@linux /]$
```

Przed znakiem zachęty występują: nazwa użytkownika, nazwa hosta oraz ciąg znaków określający bieżącą lokalizację w systemie plików. W tym przypadku:

- użytkownikiem jest *lukasz*,
- host, na którym pracujemy, to *linux*,
- katalog, w którym się znajdujemy, to / (katalog główny).

Wyświetlanie zawartości katalogu

Wyświetlanie katalogów, zwane inaczej *listowaniem ich zawartości*, można w systemie Linux wykonać za pomocą kilku poleceń. System ma bardzo rozbudowany program służący do wykonywania tego typu zadań — jest nim polecenie `ls`.

Polecenie dir

Polecenie to może przypominać jedno z poleceń systemu operacyjnego MS-DOS i wszyscy użytkownicy, którzy znają to środowisko, zapewne poczuli się jak w domu. Jednak wynik działania tego polecenia w Linuksie różni się od tego z systemu DOS. W Linuksie wyświetlana po wykonaniu tego polecenia lista plików i katalogów jest prezentowana w postaci linii, a nie w kolumnie. Trzeba przyznać, że taki zapis utrudnia nieco odczytanie drzewa katalogów i orientację w nim, szczególnie w przypadku, gdy użytkownik przyzwyczajony jest do prezentowania go w postaci znanej z MS-DOS. Oto przykład wykonania polecenia `dir` w systemie Linux:

```
[lukasz@linux /]$ dir
bin  boot  dev  etc  home  initrd  lib  lost+found  misc
   mnt  opt  proc  root  sbin  tmp  usr  var
```

Polecenie vdir

Wykonanie polecenia `vdir` powoduje wyświetlenie bardziej szczegółowych informacji o zawartości bieżącego katalogu; oprócz nazw plików i katalogów podawane są informacje o typie elementu, prawach dostępu do niego, jego właścicielu oraz kilka innych, które omówię na przykładzie polecenia `ls`. Oto przykład wykonania polecenia `vdir`:

```
[lukasz@linux /]$ vdir
drwxr-xr-x    2 root     root         4096 lis 28 17:47 bin
drwxr-xr-x    3 root     root         4096 lip  4  2003 boot
```

```
drwxr-xr-x   20 root     root       118784 lut 14 17:03 dev
drwxr-xr-x   62 root     root         4096 lut 14 17:03 etc
drwxr-xr-x    5 root     root         1024 wrz 13 21:07 home
drwxr-xr-x    2 root     root         4096 sty 25  2003
   initrd
drwxr-xr-x    9 root     root         4096 lis 28 18:06 lib
drwx------    2 root     root        16384 lip  4  2003
   lost+found
drwxr-xr-x    2 root     root         4096 sty 28  2003 misc
drwxr-xr-x    4 root     root         4096 lip  4  2003 mnt
drwxr-xr-x    2 root     root         4096 sty 25  2003 opt
dr-xr-xr-x   76 root     root            0 lut 14  2004 proc
drwxr-x---   22 root     root         4096 sty  5 21:01 root
drwxr-xr-x    2 root     root         8192 lip  4  2003 sbin
drwxrwxrwt   21 root     root         4096 lut 14 17:05 tmp
drwxr-xr-x   15 root     root         4096 lip  4  2003 usr
drwxr-xr-x   19 root     root         4096 lip  4  2003 var
```

Polecenie ls

Program ten jest bardzo rozbudowany i ma wiele parametrów, które postaram się pokrótce omówić. Wykonanie samego polecenia `ls` (bez dodatkowych parametrów) da efekt identyczny z tym, jaki powodowało wykonanie omówionego poprzednio polecenia `dir`. Aby uzyskać więcej informacji na temat zawartości katalogu, powinniśmy zastosować polecenie `ls` wraz z parametrem `-l`. Wykonanie tego polecenia w takiej postaci daje wynik identyczny z wynikiem działania polecenia `vdir`.

Jako parametr można podać ścieżkę dostępu dla katalogu, którego zawartość chcemy wyświetlić. Jeżeli jej nie podamy, zawsze wyświetlona zostanie zawartość katalogu bieżącego — tego, w którym się obecnie znajdujemy.

W zaprezentowanym tu przykładzie polecenie `ls` wywołane z parametrem `/etc` spowoduje wyświetlenie zawartości katalogu *etc*; natomiast użyte w drugiej linii przykładu polecenie `ls` wywołane bez parametru wyświetli zawartość katalogu bieżącego, którym w tym przypadku jest katalog główny.

```
[lukasz@linux /]$ ls /etc

[lukasz@linux /]$ ls
```

Najbardziej przydatnym parametrem polecenia `ls` jest `-l`; dzięki jego wykonaniu otrzymamy kompletny zestaw informacji na temat zawartości katalogu i typów zawartych w nim elementów. Zanim przedstawię resztę najważniejszych parametrów tego polecenia, wyjaśnię, jak interpretować zdobyte w ten sposób informacje. Oto przykład wykonania polecenia `ls` z parametrem `-l`:

```
[lukasz@linux /]$ ls -l
drwxr-xr-x    2 root     root         4096 lis 28 17:47 bin
drwxr-xr-x    3 root     root         4096 lip  4  2003 boot
drwxr-xr-x   20 root     root       118784 lut 14 17:03 dev
drwxr-xr-x   62 root     root         4096 lut 14 17:03 etc
drwxr-xr-x    5 root     root         1024 wrz 13 21:07 home
drwxr-xr-x    2 root     root         4096 sty 25  2003
   initrd
drwxr-xr-x    9 root     root         4096 lis 28 18:06 lib
drwx------    2 root     root        16384 lip  4  2003
   lost+found
drwxr-xr-x    2 root     root         4096 sty 28  2003 misc
drwxr-xr-x    4 root     root         4096 lip  4  2003 mnt
drwxr-xr-x    2 root     root         4096 sty 25  2003 opt
dr-xr-xr-x   76 root     root            0 lut 14  2004 proc
drwxr-x---   22 root     root         4096 sty  5 21:01 root
drwxr-xr-x    2 root     root         8192 lip  4  2003 sbin
drwxrwxrwt   21 root     root         4096 lut 14 17:05 tmp
drwxr-xr-x   15 root     root         4096 lip  4  2003 usr
drwxr-xr-x   19 root     root         4096 lip  4  2003 var
```

Wyświetlone w tym przykładzie informacje, zdobyte wskutek wykonania polecenia z parametrem `-l`, zinterpretujemy na podstawie pierwszej linii wyniku:

```
drwxr-xr-x    2 root     root         4096 lis 28 17:47 bin
```

- kolumna 1: typ elementu i prawa dostępu do niego (`drwxr-xr-x`);
- kolumna 2: liczba powiązań do tego elementu (`2`);
- kolumna 3: właściciel pliku (`root`);

- kolumna 4: grupa, która została przypisana do tego pliku (`root`);
- kolumna 5: rozmiar elementu (`4096`);
- kolumna 6: data modyfikacji (`lis 28 17:47`);
- kolumna 7: nazwa elementu (`bin`).

Rozpoznanie typu elementu

Zapis `drwxr-xr-x` z pierwszej kolumny składa się z czterech zasadniczych elementów. Pierwsza litera zawsze określa typ elementu.

Oto symbole oznaczające typy elementów:

- `-` — zwykły plik;
- `b` — specjalny plik blokowy;
- `c` — specjalny plik znakowy;
- `d` — katalog;
- `l` — dowiązanie symboliczne;
- `p` — nazwany potok;
- `s` — gniazdo.

A zatem, jak można wnioskować po zapisie `drwxr-xr-x`, rozpatrywany obiekt jest katalogiem.

Interpretacja praw dostępu

Niech przykładem, za pomocą którego wyjaśnię, na czym polega system praw dostępu w systemie Linux, będzie ten wiersz przykładowego wyniku wykonania polecenia `ls`, dotyczący katalogu *var*:

```
drwxr-xr-x   19 root     root         4096 lip  4  2003 var
```

Prawa dostępu określane są tu przez litery `r`, `w` i `x`, następujące po definiującej typ elementu literze `d` (katalog). Każda litera na odpowiedniej pozycji informuje o tym, kto i jakie prawa ma do tego pliku lub katalogu.

Zwróć uwagę na to, że w naszym przykładzie litery `x` oraz `r` występują trzykrotnie. Taki zapis określa uprawnienia według schematu: „użytkownik-grupa-inni". Litery oznaczające uprawnienia mają różne znaczenie w zależności od tego, czy stosują się do plików, czy do katalogów.

W przypadku katalogów oznaczają następujące prawa:

- `r` — do przeszukania zawartości;
- `w` — do zmiany zawartości;
- `x` — do wejścia do katalogu.

Jakie zatem uprawnienia przypisane są do katalogu *var* z naszego przykładu? Określa je następujący zapis:

```
rwx r-x r-x
```

Oznacza to, że właściciel katalogu ma prawo do jego przeszukania, zmiany jego zawartości i wejścia do katalogu, zgodnie z zapisem `rwx`.

Grupa, która została przypisana do tego elementu, ma prawa do wejścia do katalogu i przeszukania go, zgodnie z zapisem `r-x`.

Także wszyscy inni użytkownicy mają prawa do wejścia do katalogu i przeszukania go, zgodnie z zapisem `r-x`.

Jak już wspomniałem, w przypadku plików prawa dostępu określane są przez te same symbole, jednak różna jest ich interpretacja. Tym razem litery `r`, `w` i `x` oznaczają następujące prawa:

- `r` — do odczytania pliku;
- `w` — do modyfikacji pliku;
- `x` — do uruchomienia pliku.

Rozważmy przykład z następującymi prawami dostępu do pliku:

```
rw- rw- r–
```

Ten zapis informuje, iż właściciel pliku ma prawo do jego odczytywania oraz do zmiany jego zawartości, zgodnie z zapisem `rw-`.

Także grupa, która została przypisana do pliku, ma prawo do jego odczytywania i zmiany jego zawartości, zgodnie z zapisem `rw-`.

Wszyscy inni użytkownicy mają prawa jedynie do odczytania zawartości pliku, zgodnie z zapisem `r--`.

Przejdźmy teraz do omówienia kolejnych parametrów, których można użyć z poleceniem `ls`.

`-a`

Polecenie `ls` wykonane z parametrem `–a` wyświetli wszystkie pliki i katalogi w danej lokalizacji. Pokazane zostaną także pliki ukryte, które w przypadku wywołania `ls` bez tego parametru nie są widoczne. Oto przykład wykonania `ls` z parametrem `–a`; widzimy w nim także dwa symbole: „.” (kropka) i „..” (dwie kropki), które oznaczają odpowiednio katalog bieżący i nadrzędny:

```
[lukasz@linux linux]$ ls -a
.  ..  katalog  plik1  plik2  plik_kopii~  .ukryty_plik
```

`-A`

Parametr ten pozwoli zobaczyć wszystkie elementy w podanej lokalizacji, wraz z plikami ukrytymi, jednak w tym przypadku w wyniku nie będą widoczne symbole „.” i „..”, które oznaczają katalog bieżący i nadrzędny:

```
[lukasz@linux linux]$ ls -A
katalog  plik1  plik2  plik_kopii~  .ukryty_plik
```

`-B`

Użycie parametru `-B` spowoduje ukrycie plików kopii zapasowych, które znajdują się w danym katalogu. Pliki te można rozpoznać po znaku tyldy (~) występującym na końcu nazwy. Jak widać, plik *plik_kopii~* nie został tutaj pokazany:

```
[lukasz@linux linux]$ ls -B
katalog  plik1  plik2
```

`-d`

Jeżeli w danym katalogu zawarte są pliki i katalogi podrzędne, polecenie `ls` wykonane bez parametru `-d` spowoduje wyświetlenie ich wszystkich. Jeżeli jednak zastosujemy parametr `-d`, zostaną wypisane tylko elementy rozpoznane jako katalogi:

```
 [lukasz@linux linux]$ ls -d
katalog
```

`-I wzorzec, --ignore=wzorzec`

Dzięki temu parametrowi możemy nie pokazywać plików, których nazwy zawierają zdefiniowany przez nas wzorzec. Podobnie jak w systemie MS-DOS, także i tu możemy używać znaków ogólnych, takich jak gwiazdka (*), która zastępuje dowolny ciąg znaków (również pusty), oraz znak zapytania (?), który zastępuje dowolny pojedynczy znak.

W zaprezentowanym tu przykładzie została wyświetlona zawartość katalogu, z wyłączeniem tych elementów, których nazwy zaczynają się na literę „p”; zatem pominięte zostały *plik1*, *plik2* i *plik_kopii~*:

```
[lukasz@linux linux]$ ls --ignore='p*'
katalog
```

`wzorzec`

Dzięki podaniu wzorca nazwy elementu można wyświetlić tylko te znajdujące się w danym katalogu elementy, które pasują do tego wzorca. Wzorce tworzy się w taki sam sposób jak w systemie MS-DOS.

W tym przypadku chcemy, aby wyświetlane były elementy, których nazwa rozpoczyna się od litery „p":

```
[lukasz@linux linux]$ ls p*
plik1  plik2  plik_kopii~
```

W następnym przykładzie wykorzystałem także znaki zapytania, aby lepiej zobrazować ich działanie. Taki zapis pozwala na wyświetlenie wszystkich elementów, których nazwa zaczyna się od litery „p" i składa się z pięciu znaków:

```
[lukasz@linux linux]$ ls p????
plik1  plik2
```

`-R, --recursive`

Parametry te powodują rekurencyjne wyświetlenie zawartości katalogu i jego podkatalogów. W bieżącym katalogu z naszego przykładu mamy także podkatalog o nazwie *katalog*. Jak widać, po wydaniu polecenia `ls` z parametrem `-R` wyświetlona została także jego zawartość:

```
[lukasz@linux linux]$ ls -R
.:
katalog  plik1  plik2  plik_kopii~

./katalog:
plik1_w_katalogu
```

`-r, --reverse`

Użycie tych parametrów powoduje odwrócenie kolejności wyświetlania w wyniku zawartości katalogu. Najbardziej opcja ta przydaje się w przypadku sortowania, podczas którego możemy odwrócić domyślną kolejność wypisywanych elementów.

Dla większej jasności porównajmy zawartość przykładowego katalogu w porządku oryginalnym (bez opcji `-r`) i odwróconym (z użyciem tej opcji):

```
[lukasz@linux linux]$ ls
katalog  plik1  plik2  plik_kopii~

[lukasz@linux linux]$ ls -r
plik_kopii~  plik2  plik1  katalog
```

`-S, --sort=size`

Użycie tych parametrów powoduje posortowanie wyniku według wielkości plików. Największe pliki są wyświetlane jako pierwsze na liście, zaraz za katalogami:

```
[lukasz@linux linux]$ ls -S
katalog  duzy_plik  plik1  plik1.1  plik_kopii~
```

`-t, --sort=time`

Ten parametr sortuje wyniki według czasu ich modyfikacji. Najnowsze pliki są wyświetlane jako pierwsze:

```
[lukasz@linux linux]$ ls -t
duzy_plik  katalog  plik_kopii~  plik2  plik1
```

`-u, --time=atime, --time=access, --time=use`

W tym przypadku wyniki sortowane są według czasu ostatniego dostępu do pliku. Pliki, do których odwołano się ostatnio, są wyświetlane jako pierwsze.

```
[lukasz@linux linux]$ ls -u
duzy_plik  plik1  plik2  katalog  plik_kopii~
```

`-U, --sort=none`

Użycie tego parametru zapewni, że zawartość katalogu nie będzie sortowana. Elementy wyświetlone na liście wystąpią na niej dokładnie w takiej kolejności, w jakiej znajdują się w katalogu.

```
[lukasz@linux linux]$ ls -U
plik1  plik2  katalog  duzy_plik  plik_kopii~
```

`-X, --sort=extension`
Użycie tego parametru powoduje posortowanie plików według ich rozszerzeń. Pliki, które nie mają rozszerzeń, zawsze są wypisywane jako pierwsze.

```
[lukasz@linux linux]$ ls -x
duzy_plik  katalog  plik1  plik2  plik.aaa  plik.bbb
```

Przechodzenie pomiędzy katalogami

Do poruszania się w strukturze katalogów używamy polecenia `cd` wraz z parametrami, w zależności od tego, co mamy zamiar zrobić.

Najprostszym zastosowaniem tego polecenia jest użycie go bez podania jakiegokolwiek parametru. Jeżeli wpiszemy taką komendę, przejdziemy do naszego katalogu domowego:

```
[lukasz@linux lukasz]$ cd
```

Oczywiście, zamiast `lukasz` powinieneś wpisać tu nazwę swojego konta.

Innym sposobem bezpośredniego przejścia do katalogu domowego jest podanie po poleceniu `cd` znaku tyldy (~). W systemie ten znak uznawany jest za katalog domowy bieżącego użytkownika.

```
[lukasz@linux lukasz]$ cd ~
```

Jeżeli mamy zamiar przejść do wybranego przez nas katalogu, wpisujemy jego nazwę za poleceniem `cd`. Ścieżki dostępu mogą być względne, czyli podawane od miejsca, w którym się znajdujemy, lub bezwzględne, czyli podawane względem katalogu głównego, czyli /.

Dla przykładu załóżmy, że chcemy przejść z katalogu domowego do katalogu */var/www* systemu.

Pierwszą opcją jest wykonanie tego zadania krok po kroku:

```
[lukasz@linux lukasz]$ cd /
[lukasz@linux /]$ cd var
[lukasz@linux var]$ cd www
```

Druga możliwość to przejście bezpośrednio do katalogu docelowego; wystarczy podać pełną ścieżkę dostępu — względną lub bezwzględną:

```
[lukasz@linux lukasz]$ cd ../../var/www

[lukasz@linux lukasz]$ cd /var/www
```

Możemy także zastosować parametr `..` (dwie kropki), który pozwala na przejście do katalogu nadrzędnego w stosunku do tego, w którym obecnie się znajdujemy. Przypuśćmy, że obecnie jesteśmy w katalogu domowym */home/lukasz/*, a chcemy znaleźć się w katalogu nadrzędnym, czyli w */home/*. W tym celu wpisujemy polecenie `cd` z dwiema kropkami:

```
[lukasz@linux lukasz]$ cd ..
[lukasz@linux home]$
```

Tworzenie katalogów

Katalogi tworzymy za pomocą polecenia `mkdir`. Jako parametr podajemy nazwę nowego katalogu.

```
[lukasz@linux lukasz]$ mkdir katalog
```

W tym momencie utworzyliśmy katalog o nazwie *katalog*. Jeżeli jednak zamiast żądanego efektu na ekranie pojawi się komunikat:

```
mkdir: cannot create directory 'katalog': Plik istnieje
```

oznacza to, że plik lub katalog o nazwie podanej w poleceniu znajduje się już w bieżącej lokalizacji; w związku z tym musimy wymyślić inną nazwę, która jeszcze w tej lokalizacji nie występuje.

Omówmy teraz parametry polecenia `mkdir`.

`-m`

Parametr `–m` pozwala nadać odpowiednie prawa dostępu do danego katalogu w momencie jego tworzenia. Domyślnie, jeżeli nie podamy tego parametru, system sam określi prawa dostępu do katalogu.

Utwórzmy więc katalog o nazwie *kat1* najpierw bez tego parametru, a później wraz z nim:

```
[lukasz@linux linux]$ mkdir kat1
[lukasz@linux linux]$ ls -l
drwxrwxr-x    2 lukasz   lukasz       1O24 lut 21 13:45
   kat1
```

Bez omawianego parametru do katalogu zostały przypisane domyślne prawa dostępu, w postaci `drwxrwxr-x`. Teraz spróbujmy utworzyć ten sam katalog z prawami tylko do wykonania:

```
[lukasz@linux linux]$ mkdir -m 111 kat1
[lukasz@linux linux]$ ls -l
d--x--x--x    2 lukasz   lukasz       1O24 lut 21 13:44
   kat1
```

Jak widzimy, teraz prawa są ustawione zgodnie z naszymi oczekiwaniami. Oczywiście, moglibyśmy przypisać prawa dostępu do katalogu później, po jego utworzeniu, ale skoro możemy zrobić to za pomocą jednego polecenia, wybierzmy tę właśnie, wygodniejszą opcję.

`-v, --verbose`

Parametr ten wyświetla informację, czy katalog został utworzony.

```
[lukasz@linux linux]$ mkdir -v kat1
mkdir: created directory ` kat1'
```

Usuwanie katalogów

Katalogi w systemie można usunąć na dwa sposoby. Pierwszym jest użycie polecenia `rmdir`, przeznaczonego do usuwania katalogów. Utwórzmy więc katalog, który następnie usuniemy:

```
[lukasz@linux linux]$ mkdir kat1
[lukasz@linux linux]$ rmdir kat1
```

Jednak to polecenie potrafi usuwać tylko katalogi puste. Dla potwierdzenia utwórzmy nowy katalog, w nim zaś nowy plik, a następnie spróbujmy usunąć taką strukturę:

```
[lukasz@linux linux]$ mkdir kat2
[lukasz@linux linux]$ touch kat2/plik
[lukasz@linux linux]$ rmdir kat2
rmdir: `kat2': Katalog nie jest pusty
```

Jak widać, system odpowiedział, iż nie może usunąć katalogu, ponieważ nie jest on pusty. W takim przypadku potrzebujemy polecenia `rm`, które służy do usuwania plików; wraz z parametrem `-R` potrafi usunąć także katalog wraz z dowolną zawartością.

Pamiętajmy także, że nie możemy znajdować się w katalogu, który mamy zamiar usunąć.

`--ignore-fail-on-non-empty`
: Użycie tego parametru spowoduje, że nie zostaniemy poinformowani o podjętej próbie usunięcia katalogu niepustego. Skorzystajmy ponownie z utworzonego poprzednio katalogu *kat2* i spróbujmy go usunąć:

```
[lukasz@linux linux]$ rmdir kat1
[lukasz@linux linux]$ rmdir --ignore-fail-on-non-empty
   kat2
[lukasz@linux linux]$
```

Katalog ten nie został usunięty, ponieważ nie jest pusty, mimo to nie został wyświetlony komunikat o błędzie.

`--verbose`
: Jeśli użyjemy tego parametru, zostaniemy poinformowani, że katalog został pomyślnie usunięty — lub nie, jeżeli nie jest pusty.

```
[lukasz@linux linux]$ rmdir --verbose kat1
rmdir: removing directory, kat1

[lukasz@linux linux]$ rmdir --verbose kat2
rmdir: removing directory, kat2
rmdir: `kat2': Katalog nie jest pusty
```

Tworzenie plików

Pliki tworzymy zwykle za pomocą odpowiednich programów; dzięki temu każdy plik ma własny format i zawiera dane zapisane w sposób specyficzny dla programu, w którym został utworzony. Możemy jednak utworzyć także pusty plik; w tym celu należy posłużyć się poleceniem `touch`.

```
[lukasz@linux lukasz]$ touch nowy_plik
```

Usuwanie plików

Jeśli chcemy usunąć plik, powinniśmy użyć do tego polecenia `rm` wraz z nazwą pliku jako parametrem:

```
[lukasz@linux lukasz]$ rm nowy_plik
```

Po wykonaniu tego polecenia można sprawdzić, czy dany plik rzeczywiście został usunięty; wyświetlmy po prostu zawartość katalogu, w którym znajdował się usuwany plik, używając do tego polecenia `ls -l` lub `vdir`.

`-r, -R, --recursive`
: Parametry te są niezwykle przydatne, gdyż pozwalają na usuwanie rekurencyjne całych struktur na dysku. Polecenie `rm` wykonane wraz z którymś z tych parametrów pozwoli

usunąć niepuste katalogi, co było niemożliwe przy użyciu polecenia `rmdir`.

```
[lukasz@linux linux]$ rm -r katalog
```

`-f, --force`
Usuwa pliki, nie pytając o potwierdzenie, i nie zgłasza błędów w przypadku, kiedy nie może usunąć danego elementu.

```
[lukasz@linux lukasz]$ rm -f plik
[lukasz@linux lukasz]$
```

`-i, --interactive`
Powoduje wyświetlenie pytania, czy należy usunąć dany element. Odpowiedzi udzielamy przez naciśnięcie klawisza *y* w celu potwierdzenia lub jakiegokolwiek innego klawisza w celu anulowania usuwania pliku.

```
[lukasz@linux linux]$ rm -i plik1
rm: remove regular file `plik1'? y
```

`-v, --verbose`
Wyświetla informacje o pliku, który został usunięty.

```
[lukasz@linux lukasz]$ rm -v plik1
removed `plik1'
```