

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

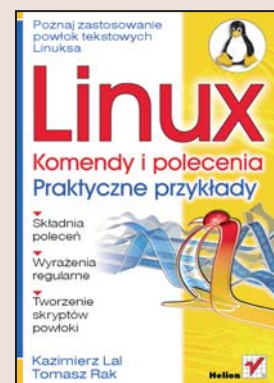
FRAGMENTY KSIĄŻEK ONLINE

Linux. Komendy i polecenia. Praktyczne przykłady

Autorzy: Kazimierz Lal, Tomasz Rak

ISBN: 83-7361-994-1

Format: B5, stron: 264



Poznaj zastosowanie powłok tekstowych Linuksa

- Składnia poleceń
- Wyrażenia regularne
- Tworzenie skryptów powłoki

Rosnąca popularność Linuksa pociągnęła za sobą rozwój środowisk graficznych, dzięki którym możliwe jest korzystanie z systemu bez jakiegokolwiek znajomości konsoli tekstowej i jej poleceń. Narzędzia oferowane przez środowiska graficzne takie jak KDE czy Gnome umożliwiają również administrowanie systemem, ale w nieco ograniczonym zakresie. Pełny dostęp do wszystkich funkcji systemu nadal możliwy jest jedynie za pośrednictwem trybu tekstowego. Jednak ilość poleceń i ich parametrów skutecznie odstrasza wielu użytkowników Linuksa od jakichkolwiek prób opanowania możliwości konsoli.

Książka „Linux. Komendy i polecenia. Praktyczne przykłady” przedstawia najczęstsze zastosowania powłoki tekstowej Linuksa. Czytając ją, poznasz typy powłok tekstowych stosowane w różnych dystrybucjach systemu Linux oraz nauczysz się korzystać z wyrażeń regularnych, filtrów i potoków. Dowiesz się, jak definiować zmienne i jak tworzyć skrypty powłoki za pomocą języka C. Znajdziesz tu również zasady administrowania systemem za pomocą poleceń wydawanych w trybie tekstowym. Dzięki zadaniom znajdującym się w książce zastosujesz poznaną wiedzę w praktyce.

- Metaznaki i wyrażenia regularne
- Stosowanie filtrów i potoków
- Definiowanie zmiennych powłoki
- Wyrażenia arytmetyczne i konstrukcje sterujące
- Tworzenie skryptów powłoki
- Zarządzanie procesami, kontami użytkowników i prawami dostępu
- Tworzenie prostych elementów graficznych

Jeśli chcesz wykorzystać pełnię możliwości Linuksa, powinieneś poznać zasady korzystania z trybu tekstowego. Dzięki tej książce zrobisz to bez problemu.



Spis treści

Wprowadzenie	7
Część I	9
Rozdział 1. Powłoki systemu Linux	11
Rozdział 2. Podstawowe polecenia powłokowe	13
2.1. Składnia poleceń	13
2.2. Metaznaki	14
2.2.1. Metaznaki *, ? i [...]	14
2.2.2. Metaznaki z listami wzorców w powłoce Korn	16
2.2.3. Metaznaki rozwijające nazwy ścieżek	17
2.3. Wyrażenia regularne	18
2.3.1. Znak kropki	20
2.3.2. Symbol \$	20
2.3.3. Symbol ^	20
2.3.4. Symbol *	20
2.3.5. Symbole [] i [^]	21
2.3.6. Symbol \{m,n\}	22
2.3.7. Symbol \(...\)	23
2.3.8. Operatory rozszerzające w egrep i awk	23
2.4. Filtry	24
2.4.1. Polecenia grep, egrep i fgrep	25
2.4.2. Polecenie cut	27
2.4.3. Polecenie sort	28
2.4.4. Polecenie uniq	29
2.4.5. Polecenie tr	30
2.4.6. Edytor strumieniowy sed	31
2.4.7. Filtr tekstowy awk	34
2.5. Przekierowanie wejścia-wyjścia	37
2.5.1. Operator >	39
2.5.2. Operator <	40
2.5.3. Operator >>	40
2.5.4. Operator <<[-]ogr	40
2.5.5. Operator 2>	41
2.5.6. Operator 2>&1	41
2.5.7. Przekierowanie w powłoce C	42
2.5.8. Zmienna noclobber	43
2.6. Potoki	43
2.6.1. Polecenie tee	45

Rozdział 3. Zmienne powłoki	47
3.1. Zmienne parametryczne	47
3.2. Zmienne środowiskowe	48
3.3. Zmienne programowe	51
3.4. Zmienne tablicowe	52
3.5. Nadzorowanie przypisywania wartości zmiennym	53
Rozdział 4. Cytowanie	57
4.1. Apostrofy	58
4.2. Cudzysłowy	58
4.3. Lewy ukośnik	60
4.4. Znaki akcentu	60
Rozdział 5. Decyzje, powtarzanie i wyrażenia arytmetyczne	63
5.1. Polecenie test	64
5.2. Polecenie expr	67
5.3. Wyrażenia arytmetyczne w powłokach Korn i Bash	69
5.3.1. Polecenie let	71
5.4. Konstrukcje sekwencyjne	72
5.5. Konstrukcje warunkowe	73
5.5.1. Konstrukcja if	73
5.5.2. Konstrukcja case	74
5.6. Konstrukcje iteracyjne	75
5.6.1. Pętla for	75
5.6.2. Pętla while	76
5.6.3. Pętla until	78
5.6.4. Polecenie xargs	78
5.7. Funkcje	79
5.8. Wyrażenia arytmetyczne i konstrukcje sterujące w powłoce C	81
5.8.1. Konstrukcja if	83
5.8.2. Konstrukcja case	84
5.8.3. Konstrukcja foreach	85
5.8.4. Pętla while	85
5.8.5. Polecenie repeat	85
Rozdział 6. Praca z wykorzystaniem powłoki	87
6.1. Określenie środowiska powłoki	87
6.2. Interaktywna praca z powłoką	89
6.2.1. Procesy i rodzaje ich pracy	89
6.2.2. Sterowanie zadaniami	90
6.2.3. Historia poleceń	96
6.3. Pisanie skryptów	102
6.3.1. Uruchamianie skryptów	104
6.3.2. Listy opcji i argumentów	105
6.3.3. Zmienne w skryptach	108
6.4. Edytor vi	110
Rozdział 7. Nadawanie praw dostępu	115
7.1. Rodzaje praw dostępu	116
7.2. Klasy praw dostępu	117
7.3. Ustawianie ochrony pliku	119
7.3.1. Polecenie chmod	119
7.3.2. Określenie domyślnych praw dostępu do plików i katalogów	122
7.3.3. Bity set-user-ID i set-group-ID	122

Rozdział 8. Grafika w powłoce	125
8.1. Okna dialogowe	125
8.1.1. Okna wyboru	126
8.1.2. Okna wiadomości	127
8.1.3. Okna informacyjne	128
8.1.4. Okna wprowadzania danych	129
8.1.5. Okna tekstowe	130
8.1.6. Okna listy wyboru	131
8.1.7. Okna postępu	135
Część II	139
Rozdział 9. Wprowadzenie do zadań	141
9.1. Ustawienie środowiska do ćwiczeń	141
9.2. Logowanie do sytemu i rozpoczęcie pracy w wierszu poleceń	143
Rozdział 10. Zadania	145
10.1. Metaznaki	145
10.2. Wyrażenia regularne i filtry	146
10.3. Przeadresowanie wejścia-wyjścia	149
10.4. Potoki i polecenie tee	150
10.5. Zmienne	150
10.6. Cytowanie	151
10.7. Polecenia test i expr — wyrażenia arytmetyczne	152
10.8. Procesy i sterowanie zadaniami	152
10.9. Historia poleceń	153
10.10. Pisanie skryptów i konstrukcje sterujące	154
10.11. Edytor vi	156
10.12. Prawa dostępu do plików i katalogów	156
10.13. Grafika w powłoce	158
Rozdział 11. Odpowiedzi do zadań	161
11.1. Metaznaki	161
11.2. Wyrażenia regularne i filtry	165
11.3. Przeadresowanie wejścia-wyjścia	182
11.4. Potoki i polecenie tee	185
11.5. Zmienne	188
11.6. Cytowanie	192
11.7. Polecenia test i expr — wyrażenia arytmetyczne	194
11.8. Procesy i sterowanie zadaniami	197
11.9. Historia poleceń	198
11.10. Pisanie skryptów i konstrukcje sterujące	206
11.11. Edytor vi	213
11.12. Prawa dostępu do plików i katalogów	219
11.13. Grafika w powłoce	226
Dodatki	249
Bibliografia	251
Skorowidz	253

Rozdział 4.

Cytowanie

Domyślnym działaniem powłoki jest obliczanie wartości i podstawianie. Wiersz poleceń jest przeglądany, spacje dzielą go na słowa, napotkane znaki specjalne wskazują, jakie czynności mają być wykonane, a wyniki są wstawiane w odpowiednie miejsce wiersza. Nie zawsze jednak takie działanie powłoki jest pożądane [1] [11] [13] [18]. Może się zdarzyć, że w pewnych sytuacjach powłoka zinterpretuje wprowadzone komendy niezgodnie z intencją użytkownika. Można się przed tym zabezpieczyć, odbierając niektórym znakom (traktowanym przez powłokę w sposób szczególny) ich specjalne znaczenie. Służy do tego mechanizm cytowania znaków (ang. *quoting*). Za pomocą cytowania chroni się spacje występujące w wierszu poleceń (cytowany ciąg znaków jest interpretowany jak pojedyncze słowo) oraz nadzoruje rozwinięcia (w szczególności nazw plików i zmiennych). Znaki cytowania rozróżniane przez powłokę zostały przedstawione w tabeli 4.1.

Tabela 4.1. *Znaki cytowania*

Znak(i)	Opis
Brak cytowania	Brak cytowania oznacza, że wszystkie słowa wydzielone za pomocą spacji są interpretowane przez powłokę.
Apostrofy 'ciag_znakow'	Tekst zawarty w apostrofach jest chroniony. Cały <i>ciag_znakow</i> jest przesyłany jako argument do polecenia.
Cudzysłowy "ciag_znakow "	Tylko znaki \$, \ oraz znak akcentu ` mają specjalne znaczenie w <i>ciagu_znakow</i> . Wszystkie inne znaki są chronione. Cudzysłów zapewnia jednoznaczną interpretację ciągu znaków, nawet w przypadku użycia znaku spacji.
Lewy ukośnik \c	Znak występujący po \ jest chroniony przed interpretacją przez powłokę.
Akcenty `polecenie`	<i>Polecenie</i> umieszczone w akcentach jest wykonywane przez podpowłoki i w jego miejsce podstawiany jest uzyskany wynik. Analiza dokonywana przez podpowłoki przebiega w zwykły sposób. Brane są pod uwagę włączone znaki cytowania.

4.1. Apostrofy

Apostrofy są wygodnym sposobem na to, by ciąg znaków pozbawić specjalnego znaczenia. Znaki znajdujące się pomiędzy otwierającym i zamykającym apostrofem stają się znakami zwykłymi. Ponadto apostrofy łączą tekst w jeden argument. Spacja nie jest wtedy separatorem argumentów.

Przykład 4.1.

Apostrofy użyte w poniższym poleceniu `grep` sprawiają, że ciąg znaków `zapisz wiadomosc` potraktowany zostanie jako jeden argument wyszukiwany w pliku `dane`:

```
grep 'zapisz wiadomosc' dane
```

Jeżeli nie użylibyśmy znaków apostrofu i zapisalibyśmy polecenie `grep` w takiej postaci:

```
grep zapisz wiadomosc dane
```

słowo *zapisz* szukane byłoby w plikach *wiadomosc* i *dane*.

Przykład 4.2.

Założmy, że wykonana została sekwencja poleceń:

```
zmienna='wartosc wpisana do zmiennej'  
echo $zmienna
```

Wynikiem będzie wyświetlenie na ekranie ciągu znaków:

```
wartosc wpisana do zmiennej
```

Gdy użyjemy polecenia:

```
echo '$zmienna'
```

znak `$` utraci specjalne znaczenie i na ekranie otrzymamy:

```
$zmienna
```

W analogiczny sposób swoje specjalne znaczenie tracą wszystkie inne znaki specjalne. Łatwo możemy się domyślić, co by się wydarzyło, gdybyśmy nie zastosowali apostrofów.

4.2. Cudzysłowy

Cudzysłowy działają podobnie jak apostrofy, ale nie pozbawiają wszystkich znaków specjalnych ich znaczenia. Znaki `$`, `\` oraz ``` utrzymują swoje specjalne znaczenie w ciągu znaków. W cudzysłowach możliwe jest podstawianie wartości zmiennych, ale podstawianie nazw plików nie jest już możliwe. Podstawianie wykonywane na zmiennych umieszczonych wewnątrz cudzysłowów różni się nieco od zwykłego podstawiania w wierszu poleceń. Wynika to z ochrony zapewnianej przez cudzysłowy.

Przykład 4.3.

Przeanalizujemy skrypt:

```
$ zmienna="Ochrona spacji zawartych w tekście"
$ echo $zmienna
Ochrona spacji zawartych w tekście
$ echo "$zmienna"
Ochrona spacji zawartych w tekście
```

Umieszczenie zmiennej w cudzysłowie powoduje ochronę spacji w niej umieszczonych. Normalnie powłoka przetwarza zmienną w ten sposób, że usuwa wszystkie nadmiarowe spacje i każdy element traktuje oddzielnie. W naszym przykładzie polecenie:

```
$ echo $zmienna
```

usuwa spacje poprzedzające poszczególne słowa, pobiera te słowa jako oddzielne argumenty i wyświetla, rozdzielając je pojedynczymi spacjami. Gdy mamy do czynienia z zapisem w cudzysłowach:

```
$ echo "$zmienna"
```

spacje są chronione i polecenie echo otrzymuje cały ciąg znaków jako pojedynczy argument.

Przykład 4.4.

Rozważmy teraz przykład obrazujący różnice między użyciem apostrofów, cudzysłów lub nieużyciem żadnego z tych znaków. Zakładamy, że w katalogu bieżącym istnieje plik *tekst* oraz istnieje zmienna *zm* o wartości napis.

```
$ echo Plik tekst zawiera $zm
Plik tekst zawiera napis
$echo 'Plik teks* zawiera $zm'
Plik teks* zawiera $zm
$echo "Plik teks* zawiera $zm"
Plik teks* zawiera napis
```

Apostrofy pozbawiają specjalnego znaczenia znaki * i \$, a cudzysłów pozbawia specjalnego znaczenia tylko znak *.

Przykład 4.5.

Załóżmy, że chcemy za pomocą polecenia *grep* wyszukać wiersze w pliku *abonenci* według wzorca zawartego w zmiennej *dane* o wartości Jan Kowalski. Polecenie:

```
grep $dane abonenci
```

po wstawieniu wartości zmiennej będzie miało postać:

```
grep Jan Kowalski abonenci
```

i spowoduje, że *grep* będzie szukał wyrazu *Jan* w plikach *Kowalski* i *abonenci*. Dopiero zastosowanie cudzysłówów:

```
grep "$dane" abonenci
```

pozwoli uzyskać polecenie:

```
grep "Jan Kowalski" abonenci
```

i da pożądany efekt.

4.3. Lewy ukośnik

Lewy ukośnik `\` służy do pozbawienia specjalnego znaczenia znaku, który występuje za nim. Można w ten sposób chronić dowolny znak w dowolnym miejscu. Znak `\` jest często używany wewnątrz cudzysłowów do ochrony następujących znaków cudzysłowu, na przykład:

```
$ echo "Powiesc pt. \"Lałka\""  
Powiesc pt. "Lałka"
```

Innym zastosowaniem znaku `\` jest ochrona przed podstawianiem przez powłokę nazw plików lub wartości zmiennych. W tym celu za jego pomocą należy „zasłonić” znaki specjalne:

```
$ echo To chyba za drogo\  
To chyba za drogo?
```

Znak lewego ukośnika jest więc wygodnym narzędziem, które bardzo ułatwia nadzorowanie podstawień wykonywanych przez powłokę.

4.4. Znaki akcentu

Znaki akcentu ``` powodują, że powłoka uważa wszystko, co znajduje się pomiędzy nimi, za polecenie. Dzięki temu są one bardzo użyteczne przy pisaniu skryptów i programów, a szczególnie przydają się do przypisywania wartości zmiennym. Pomiedzy apostrofami można umieścić dowolne polecenie lub zestaw poleceń. Każde z nich będzie wykonywane, jakby zostało wprowadzone w wierszu poleceń. Taki efekt bierze się stąd, że każde polecenie jest wykonywane we własnej podpowłoce. Oznacza to, że polecenie takie nie będzie pamiętane po powrocie do powłoki, z której zostało wywołane.

Przykład 4.6.

Przeanalizujmy skrypt:

```
$ data=date  
$ echo $data  
date  
$ data=`date`  
$ echo $data  
Mon May 29 19:58:29 MDT 2005
```


Początkowo zmiennej *data* został przypisany ciąg znaków *date*. Powłoka uznała wszystkie znaki po prawej stronie znaku równości za część stałej, która ma stanowić wartość zmiennej. W kolejnym poleceniu użyte zostały znaki akcentu:

```
$ data=`date`
```

dzięki czemu wynik polecenia *date* zastąpił samo polecenie. Ostatecznie zmiennej *data* została przypisana bieżąca data.

Przykład 4.7.

Aby wyświetlić informację o liczbie plików w bieżącym katalogu, można użyć polecenia:

```
$ echo "Jest `ls|wc -l` pliki w `pwd`"
```

Znaki ``` spowodują podstawienie wartości w miejsce poleceń `ls|wc -l` i `pwd`, co przykładowo może dać:

```
Jest 23 pliki w /home/jan
```