

*Niezbędny pomocnik
każdego użytkownika Linuksa!*

Wydanie II

Leksykon kieszonkowy

LINUX



O'REILLY

Daniel J. Barrett

Tytuł oryginału: Linux Pocket Guide, Second Edition

Tłumaczenie: Adam Bąk

ISBN: 978-83-246-5602-8

© 2012 Helion S.A.

Authorized Polish translation of the English edition of Linux Pocket Guide, 2nd Edition ISBN 9781449316693 © 2012 Daniel Barrett.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/link2>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Co zawiera ta książka?	5
Gdzie szukać pomocy?	10
Czym jest Linux?	12
System plików	17
Powłoka	26
Podstawowe operacje na plikach	41
Operacje na katalogach	45
Przeglądanie plików	48
Tworzenie i edytowanie plików	58
Właściwości plików	63
Lokalizacja plików	74
Manipulowanie plikami tekstowymi	82
Kompresowanie i pakowanie plików	94
Porównywanie plików	99
Drukowanie	104
Sprawdzanie pisowni	106
Dyski i systemy plików	107
Kopie bezpieczeństwa i zdalne przechowywanie	112
Przeglądanie procesów	117
Kontrola procesów	122
Planowanie zadań	125

Logowanie, wylogowanie i kończenie pracy.....	130
Użytkownicy i ich środowisko	131
Zarządzanie kontami użytkowników	135
Jak zostać superużytkownikiem?.....	139
Zarządzanie grupami	141
Informacje o komputerze	143
Umieszczenie komputera.....	146
Połączenia sieciowe	150
Poczta elektroniczna.....	154
Przeglądanie stron WWW	159
Grupy dyskusyjne	163
Komunikatory.....	165
Pisanie na ekranie	167
Obliczenia matematyczne	172
Czas i data	175
Grafika i wygaszacze ekranu	178
Audio	182
Wideo.....	185
Instalowanie oprogramowania.....	187
Programowanie skryptów powłoki.....	191
Postowie	204
Skorowidz.....	205

Powłoka

Aby mieć możliwość wydawania Linuksowi poleceń, będziemy potrzebowali miejsca, w którym można by je wpisywać. To miejsce nazywane jest *powłoką*, która w Linuksie spełnia rolę interfejsu użytkownika wiersza poleceń. Polecenia są w niej wpisywane, a po naciśnięciu klawisza *Enter* powłoka uruchamia program (lub programy) podany w powłoce (informacje o tym, jak uruchomić powłokę, zamieszczono w sekcji „Uruchamianie powłoki”).

Na przykład w celu sprawdzenia, kto jest zalogowany do systemu, można w powłoce uruchomić następujące polecenie:

```
$ who
kowalski      :0      Wrz 23 20:44
nowak        pts/0    Wrz 15 12:51
malinowski   pts/1    Wrz 22 21:15
kowalski     pts/2    Wrz 22 21:18
```

Znak dolara jest znakiem zachęty powłoki i oznacza, że jest ona gotowa do pracy i czeka na wprowadzenie polecenia. Pojedyncze polecenie może też uruchomić jednocześnie wiele programów, a nawet połączyć je w taki sposób, aby ze sobą współpracowały. Oto polecenie przekierowujące wyjście programu *who* i podłączające je do wejścia programu *wc*, który zajmuje się zliczaniem wierszy w pliku tekstowym. W efekcie otrzymamy liczbę wierszy w tekście wygenerowanym przez program *who*:

```
$ who | wc -l
4
```

informującą, ile osób jest zalogowanych w systemie⁵. Połączenie między programami *who* i *wc* wykonuje pionowa kreseczka, która nazywana jest *potokiem*.

⁵ Tak naprawdę dowiemy się, ile powłok zostało uruchomionych przez użytkowników. Jeżeli jeden użytkownik będzie miał uruchomionych kilka powłok, jak w naszym przykładzie użytkownik *kowalski*, na liście pojawi się on kilka razy.

Sama powłoka tak naprawdę również jest zwykłym programem, a w Linuksie dostępnych jest kilka różnych powłok. W tej książce skoncentrujemy się na powłoce bash (*Bourne-Again Shell*), która znajduje się w katalogu `/bin/bash` i która w systemach Linux najczęściej jest powłoką domyślną.

Powłoka a programy

Polecenie w momencie uruchomienia może wywołać jakiś program (na przykład `who`), ale może też ono być tak zwanym *poleceniem wbudowanym* (ang. *built-in command*), czyli funkcją wykonywaną przez samą powłokę. Takie polecenia można odróżniać za pomocą polecenia `type`:

```
$ type who
who is /usr/bin/who
$ type cd
cd is shell builtin
```

Dobrze jest wiedzieć, jakie funkcje udostępnia sama powłoka, a jakie są częścią Linuksa. W kilku kolejnych punktach będziemy opisywać wyłącznie funkcje powłoki.

Wybrane funkcje powłoki bash

Powłoka ma znacznie więcej funkcji niż tylko umożliwienie uruchomienia poleceń. Potrafi też bardzo ułatwić tę pracę: nazwy wieloznaczne (*wildcards*), pozwalające na dopasowanie nazw plików; „historia poleceń” do szybkiego przywoływania wykonanych uprzednio poleceń; potoki umożliwiające połączenie wyjścia jednego programu z wejściem drugiego; zmienne przechowujące wartości wykorzystywane przez powłokę i wiele innych funkcji. Poświęcenie nieco czasu na naukę tych funkcji pozwoli szybciej i wydajniej pracować w Linuksie. Ta krótka prezentacja przedstawiła zaledwie ułamek możliwości powłoki, czas dowiedzieć się więcej (pełną dokumentację powłoki można wyświetlić za pomocą polecenia `info bash`).

Nazwy wieloznaczne

Nazwy wieloznaczne to swego rodzaju skrót do grupy plików o podobnych nazwach. Na przykład `a*` oznacza wszystkie pliki, których nazwy zaczynają się od litery „a”. Nazwy wieloznaczne są „rozwijane” przez powłokę w nazwy rzeczywistych plików, pasujących do podanego wzorca. Wobec czego, jeżeli wpisujemy polecenie:

\$ ls a*

powłoka najpierw rozwinie parametr `a*` w nazwy plików rozpoczynających się na literę „a”, znajdujących się w aktualnym katalogu roboczym, tak jakby zostało wpisane polecenie:

\$ ls andrychow ametyst agrest

Polecenie `ls` nigdy nie dowie się, że użytkownik stosował nazwy wieloznaczne, zawsze otrzyma tylko listę nazw plików przygotowaną przez powłokę. Ważne, aby zapamiętać, że każde polecenie Linuksa, niezależnie od jego pochodzenia, może być używane z nazwami wieloznacznymi i innymi funkcjami powłoki shell.

W nazwach wieloznacznych nigdy nie wolno stosować dwóch znaków: myślnika na początku oraz ukośnika (`/`); znaki te muszą być zawsze używane w ich oryginalnym znaczeniu, co oznacza, że do wyszukania pliku o nazwie `.profile` trzeba użyć nazwy wieloznacznej `.pro*`, a aby wyświetlić wszystkie pliki w katalogu `/etc` z końcówką `conf`, należy wpisać `/etc/*conf`.

Pliki z kropką

Pliki, których nazwa jest poprzedzona kropką, zwane po prostu plikami z kropką, to specjalne pliki w Linuksie. Takie pliki nie są wyświetlane w niektórych programach:

- polecenie `ls` nie wyświetli ich, chyba że zostanie wykonane z opcją `-a`;
- nazwy wieloznaczne powłoki również ich nie używają.

W praktyce oznacza to, że są one niewidoczne do czasu, aż użytkownik wprost nie zażąda ich wyświetlenia. Dlatego czasem określa się je jako „ukryte pliki”.

Nazwa wieloznaczna	Znaczenie
<code>*</code>	Zero lub więcej kolejnych znaków
<code>?</code>	Dowolny pojedynczy znak
<code>[zbiór]</code>	Dowolny pojedynczy znak z podanego <i>zbioru</i> . Najczęściej zapisywane są tutaj sekwencje znaków, takie jak <code>[aeiouAEIOU]</code> , oznaczająca wszystkie samogłoski, lub zakresy definiowane za pomocą myślnika, takie jak <code>[A-Z]</code> , oznaczający wszystkie wielkie litery
<code>^[zbiór]</code>	Dowolny pojedynczy znak <i>nie</i> znajdujący się w podanym <i>zbiorze</i> (wcześniejszy przykład)
<code>![zbiór]</code>	To samo, co <code>^[zbiór]</code> .

Jeżeli w podawanym zbiorze znaków chcielibyśmy umieścić znak myślніка, trzeba zapisać go jako pierwszy lub ostatni znak zbioru. Aby w zbiorze umieścić znak zamykającego nawiasu kwadratowego (]), należy umieścić go na pierwszej pozycji w zbiorze. Aby w zbiorze umieścić znak ^ lub wykrzyknika (!), nie należy umieszczać go na pierwszej pozycji w zbiorze.

Rozwijanie nawiasów klamrowych

Podobnie jak nazwy wieloznaczne, wyrażenia zamknięte w nawiasach klamrowych również rozwijane są w wiele parametrów przekazywanych poleceniom. Wyrażenie rozdzielane przecinkami:

```
{X,YY,ZZZ}
```

zostanie rozwinięte najpierw do X, następnie do YY, a w końcu do ZZZ w postaci:

```
$ echo kan{X,Y,ZZZ}apka
kanXapka kanYapka kanZZapka
```

Nawiasy klamrowe można stosować z dowolnymi ciągami znaków, natomiast nazwy wieloznaczne można stosować tylko do określania nazw plików. Powyższy przykład działa niezależnie od tego, jakie pliki znajdują się w aktualnym katalogu roboczym.

Zmienne powłoki

Zmienne powłoki i ich wartości można zdefiniować poprzez operację przypisania:

```
$ MOJAZMIENNA=3
```

Aby odwołać się do wartości przechowywanej w zmiennej, wystarczy umieścić znak dolara przed nazwą zmiennej:

```
$ echo $MOJAZMIENNA
3
```

Niektóre zmienne są standardowe i najczęściej definiowane są przez samą powłokę w momencie logowania do systemu.

Zmienna	Znaczenie
DISPLAY	Nazwa wyświetlacza systemu okien X
HOME	Nazwa katalogu domowego użytkownika, na przykład <i>/home/kowalski</i>
LOGNAME	Nazwa użytkownika stosowana w czasie logowania, na przykład <i>kowalski</i>

Zmienna	Znaczenie
MAIL	Ścieżka do katalogu z pocztą przychodzącą, na przykład <code>/var/spool/mail/kowalski</code>
OLDPWD	Poprzedni katalog roboczy powłoki, używany przed ostatnim poleceniem <code>cd</code>
PATH	Ścieżka przeszukiwania powłoki; katalogi są rozdzielane dwukropkami
PWD	Aktualny katalog roboczy powłoki
SHELL	Ścieżka do powłoki użytkownika, na przykład <code>/bin/bash</code>
TERM	Typ wykorzystywanego terminala, na przykład <code>xterm</code> lub <code>vt100</code>
USER	Nazwa użytkownika

Aby zobaczyć wszystkie zmienne powłoki, należy wydać polecenie:

```
$ printenv
```

Zakres zmiennej (to znaczy zbiór programów, które wiedzą o jej istnieniu) domyślnie definiowany jest jako powłoka, w której została ona zdefiniowana. Aby udostępnić zmienną i jej wartość innym programom wywoływanym z powłoki (na przykład podpowłokom), należy użyć polecenia `export`:

```
$ export MOJAZMIENNA
```

albo jej skrótu:

```
$ export MOJAZMIENNA=3
```

Taka zmienna nazywana jest wtedy zmienną środowiskową, ponieważ dostępna jest też z innych programów w „środowisku” powłoki. W powyższym przykładzie wyeksportowana zmienna `MOJAZMIENNA` jest dostępna dla wszystkich programów działających w tej samej powłoce (dotyczy to również skryptów powłoki; więcej informacji w sekcji „Zmienne”).

Aby pewnemu programowi jednokrotnie udostępnić wartość zmiennej, wystarczy nadać jej tymczasową wartość, wpisując `zmienna=wartość` w powłoce tuż przed samym poleceniem:

```
$ echo $HOME
/home/kowalski
$ HOME=/home/nowak echo "Mój katalog domowy to $HOME"
Mój katalog domowy to /home/nowak
$ echo $HOME
/home/kowalski
```

Oryginalna wartość zmiennej nie zmienia się

Ścieżka wyszukiwania

Programy są porozrzucane po całym systemie plików Linuksa, katalogach takich jak `/bin` czy `/usr/bin`. W jaki sposób powłoka odnajduje program uruchamiany z linii poleceń? Krytyczne znaczenie ma zmienna `PATH`, która przekazuje powłoce informacje, gdzie można znaleźć programy. Gdy wpisujemy w wierszu poleceń dowolne polecenie:

```
$ who
```

powłoka musi odnaleźć program `who` w katalogach Linuksa. Sprawdza więc zawartość zmiennej `PATH`, w której zapisana jest sekwencja katalogów rozdzielanych dwukropkami:

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/home/kowalski/bin
```

Następnie powłoka sprawdza, czy program `who` nie znajduje się w jednym z katalogów zapisanych w zmiennej `PATH`. Jeżeli powłoce uda się znaleźć program `who` (na przykład `/usr/bin/who`), wtedy zostanie on uruchomiony. W przeciwnym wypadku wyświetlony zostanie jedynie komunikat:

```
bash: who: command not found
```

Aby tymczasowo dodać katalogi do ścieżki wyszukiwania powłoki, można zmodyfikować zmienną `PATH`. Na przykład poniższe polecenie do ścieżki wyszukiwania dodaje katalog `/usr/sbin`:

```
$ PATH=$PATH:/usr/sbin
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/home/kowalski/bin:/usr/sbin
```

Zmiana zostanie wprowadzona tylko w bieżącej powłoce. Aby wprowadzić ją na stałe do ścieżki wyszukiwania, należy zmodyfikować wartość zmiennej `PATH` w pliku `~/.bash_profile`, co zostanie opisane w podrozdziale „Dostosowywanie zachowań powłoki”. Trzeba się tylko wylogować i ponownie zalogować.

Aliasy

Wbudowane polecenie `alias` pozwala na definiowanie wygodnych skrótów dłuższych poleceń, co wydatnie zmniejsza liczbę wpisywanych znaków. Na przykład polecenie:

```
$ alias ll='ls -l'
```

definiuje nowe polecenie ll, które uruchamia polecenie ls -l.

```
$ ll
razem 436
-rw-r--r--    1 kowalski   3584 paź 11 14:59 plik1
-rwxr-xr-x    1 kowalski    72 sie  6 23:04 plik2
...
```

Aby aliasy były dostępne po każdym logowaniu do systemu, należy je zdefiniować w pliku `~/bashrc` (proszę zobaczyć podrozdział „Dostosowywanie zachowań powłoki”)⁶. Aby wypisać wszystkie zdefiniowane aliasy, należy uruchomić polecenie `alias`. Jeżeli aliasy nie wydają się być wystarczającym narzędziem (nie przyjmują parametrów i nie pozwalają na wykonywanie rozgałęzień), proszę przeczytać podrozdział „Programowanie skryptów powłoki”, uruchomić polecenie `info bash` i przeczytać sekcję „FUNCTIONS”.

Przekierowanie wejścia i wyjścia

W powłoce możliwe jest przekierowanie standardowego wejścia, standardowego wyjścia i standardowego strumienia błędów programów do i z plików. Innymi słowy, za pomocą operatora `<` można spowodować, że każdy program, odczytujący dane ze standardowego wejścia, może je otrzymać z podanego pliku:

```
$ moje polecenie < plik_wejscowy
```

Podobnie każde polecenie, zapisujące wyniki do standardowego wyjścia, może zapisywać je do podanego pliku:

```
$ moje polecenie > plik_wyjsciowy      Utworzenie lub nadpisanie pliku wyjściowego
$ moje polecenie >> plik_wyjsciowy      Dopisanie danych na końcu pliku
```

Polecenie zapisujące cokolwiek do standardowego strumienia błędów również może zostać przekierowane tak, żeby zapisywać do pliku, podczas gdy dane na standardowym wyjściu wciąż będą wyświetlane na ekranie:

```
$ moje polecenie 2> plik_bledow
```

Aby przekierować do plików zarówno strumień błędów, jak i standardowe wyjście:

```
$ moje polecenie > plik_wyjsciowy 2> plik_bledow  Dwa osobne pliki
$ moje polecenie > plik_wyjsciowy 2>&1             Do tego samego pliku
```

⁶ W niektórych konfiguracjach wykorzystuje się do tego osobny plik `~/bash_aliases`.

Potoki

Za pomocą operatora potoku (`|`) możliwe jest takie przekierowanie standardowego wyjścia jednego programu, aby stało się ono standardowym wejściem innego. Na przykład polecenie:

```
$ who | sort
```

przesyła wyjście programu `who` do wejścia programu `sort`, wypisując posortowaną listę zalogowanych w systemie użytkowników. Potoków można używać wielokrotnie. Dane wyjściowe polecenia `who` zostały posortowane (polecenie `sort`). Następnie za pomocą programu `awk` został wyodrębniony pierwszy wiersz, a zastosowanie polecenia `less` pozwoliło wyświetlić te dane na pojedynczej stronie (ekranie):

```
$ who | sort | awk '{print $1}' | less
```

Łączenie poleceń

Aby w wierszu poleceń wywołać kilka poleceń jedno za drugim, należy rozdzielać je znakiem średnika (`;`):

```
$ polecenie1 ; polecenie2 ; polecenie3
```

Aby wykonać powyższą sekwencję poleceń, a jednocześnie zatrzymać wykonywanie kolejnych, jeżeli jedno z nich nie wykona się prawidłowo, należy polecenia rozdzielać znakami `&&`:

```
$ polecenie1 && polecenie2 && polecenie3
```

Aby wykonać powyższą sekwencję poleceń, a jednocześnie zatrzymać wykonywanie kolejnych, jeżeli jedno z nich wykona się prawidłowo, należy polecenia rozdzielać znakami `||`:

```
$ polecenie1 || polecenie2 || polecenie3
```

Cytowanie

Powłoka traktuje standardowo znaki białe jako znaki rozdzielające słowa w wierszu poleceń. Jeżeli polecenie wymaga, aby któreś ze słów zawierało w sobie takie znaki (na przykład nazwa pliku zawierająca w sobie spacje), należy zamknąć je wewnątrz pojedynczych lub podwójnych cudzysłowów. Tak oznaczone słowa powłoka będzie traktować jako niepodzielną całość. Słowa zamknięte w pojedynczych cudzysłowach nie ulegają żadnym zmianom, natomiast słowa zamknięte w podwójnych cudzysłowach pozwalają na przetworzenie zawartych w nich konstrukcji powłoki, takich jak zmienne.

```
$ echo 'Zmiennej HOME przypisano wartość $HOME'
Zmiennej HOME przypisano wartość $HOME
$ echo "Zmiennej HOME przypisano wartość $HOME"
Zmiennej HOME przypisano wartość /home/kowalski
```

Cudzysłowy odwrócone powodują, że zawarty w nich tekst traktowany jest jako polecenie powłoki; tekst ten zamieniany jest standardowym wyjściem tego polecenia:

```
$ /usr/bin/whoami      Program wyświetli nazwę użytkownika
kowalski
$ echo Nazywam się ~/usr/bin/whoami`
Nazywam się kowalski
```

Znaki specjalne

Jeżeli znak ma znaczenie specjalne, ale istnieje potrzeba użycia bezpośredniego (na przykład znaku gwiazdki (*)) jako rzeczywistej gwiazdki, a nie znaku nazwy wieloznacznej), należy poprzedzić go znakiem ukośnika (\).

```
$ echo a*      Jako nazwa wieloznaczna, nazwy plików rozpoczynające się literą „a”
andrychow ametyst agrest
$ echo a\*      Jako bezpośredni znak gwiazdki
a*
$ echo "Po angielsku DOM to $HOME"      Znak dolara oznacza wartość zmiennej
Po angielsku DOM to /home/kowalski
$ echo "Po angielsku DOM to \$HOME"      Po prostu symbol dolara
Po angielsku DOM to $HOME
```

Podobny mechanizm można stosować ze znakami sterującymi (tabulacje, nowe wiersze, ^D itd.), aby stosować je w wierszu poleceń. Wystarczy umieścić przed nimi znak ^V. Ta metoda przydaje się szczególnie w przypadku znaków tabulacji (^I), które normalnie zostałyby użyte przez powłokę do uzupełnienia nazwy pliku (zobacz podrozdział „Uzupełnianie nazw plików”).

```
$ echo "Tabulacja rozciąga się odtąd^V^Idotąd"
Tabulacja rozciąga się odtąd      dotąd
```

Edycja w wierszu poleceń

Powłoka bash pozwala na edytowanie wiersza poleceń za pomocą kombinacji klawiszy wywodzących się z edytorów emacs i vi (proszę zobaczyć podrozdział „Tworzenie i edytowanie plików”). Aby włączyć w edycji wiersza poleceń kombinacje klawiszy edytora emacs, należy uruchomić następujące polecenie (umieszczenie go w pliku ~/.bash_profile uruchomi ten tryb na stałe):

```
$ set -o emacs
```

Aby uzyskać kombinacje klawiszy edytora `vi`, należy uruchomić polecenie:

```
$ set -o vi
```

Kombinacje klawiszy edytora Emacs	Kombinacje klawiszy edytora vi (najpierw należy nacisnąć klawisz ESC)	Znaczenie
<code>^P</code> lub klawisz strzałki w górę	<code>k</code>	Powrót do poprzedniego wiersza polecenia
<code>^N</code> lub klawisz strzałki w dół	<code>j</code>	Przejdźcie do następnego wiersza polecenia
<code>^F</code> lub klawisz strzałki w prawo	<code>l</code>	Przejdźcie o jeden znak do przodu
<code>^B</code> lub klawisz strzałki w lewo	<code>h</code>	Przejdźcie o jeden znak do tyłu
<code>^A</code>	<code>O</code>	Przejdźcie na początek wiersza
<code>^E</code>	<code>\$</code>	Przejdźcie na koniec wiersza
<code>^D</code>	<code>x</code>	Usunięcie następnego znaku
<code>^U</code>	<code>^U</code>	Usunięcie całego wiersza

Historia poleceń

Możliwe jest przywołanie poprzednio uruchomionych poleceń (czyli *historii* powłoki), edytowanie ich i ponowne uruchomienie. Poniżej podano kilka przydatnych poleceń związanych z historią powłoki.

Polecenie	Znaczenie
<code>history</code>	Wypisuje historię powłoki
<code>history N</code>	Wypisuje ostatnich <i>N</i> poleceń z historii powłoki
<code>history -c</code>	Usuwa historię powłoki
<code>!!</code>	Ponownie uruchamia poprzednie polecenie
<code>!N</code>	Ponownie uruchamia polecenie numer <i>N</i> w historii powłoki
<code>!-N</code>	Ponownie uruchamia polecenie wpisane <i>N</i> poleceń wcześniej
<code>!\$</code>	Ostatni parametr poprzedniego polecenia; doskonale nadaje się do sprawdzania, przed usuwaniem plików, czy one rzeczywiście istnieją: <pre>\$ ls a* acorn.txt affidavit \$ rm !\$</pre>

Polecenie	Znaczenie
!* \$ ls a b c a b c \$ wc !* 103 252 2904 a 12 25 384 b 25473 65510 988215 c 25588 65787 991503 total	Wszystkie parametry poprzedniego polecenia

Uzupełnianie nazw plików

Po naciśnięciu klawisza *TAB* w czasie wpisywania polecenia w wierszu poleceń powłoka automatycznie uzupełni wpisywaną nazwę pliku. Jeżeli do wpisanej do tej pory części nazwy pasuje kilka nazw plików, powłoka sygnałem dźwiękowym poinformuje o tej wieloznaczności. Szybkie ponowne naciśnięcie klawisza *TAB* spowoduje, że powłoka wypisze na ekranie wszystkie dostępne nazwy plików. Proszę spróbować uruchomić takie polecenia:

```
$ cd /usr/bin
$ ls un<TAB><TAB>
```

Powłoka wyświetli wszystkie pliki w katalogu */usr/bin/* rozpoczynające się od *un*, takie jak *unique*, *units*, *unzip*. Aby zawęzić wyszukiwanie, należy dodać jeszcze kilka znaków i ponownie wcisnąć klawisz *Tab*.

Kontrola zadań powłoki

jobs	Wypisuje listę zadań.
&	Uruchamia zadanie w tle.
^Z	Wstrzymuje działanie aktualnego zadania.
suspend	Wstrzymuje działanie powłoki.
fg	Wznowienie zadania i przeniesienie go na pierwszy plan.
bg	Sprawia, że wstrzymane zadanie wznawia pracę w tle.

Wszystkie powłoki w Linuksie posiadają mechanizm *kontroli zadań*, czyli możliwość uruchamiania programów w tle (działają one bez kontaktu z ekranem i klawiaturą) i na pierwszym planie (aktywny proces podłączony do ekranu i klawiatury). *Zadanie* to po prostu część pracy powłoki. W momencie interaktywnego uruchomienia polecenia

powłoka zaczyna traktować je jak kolejne zadanie. Po zakończeniu działania przez polecenie, związane z nim zadanie znika. Zadania to pojęcie wyższego poziomu niż procesy systemu Linux; sam system w ogóle nie zajmuje się zadaniami, są one konstrukcjami stosowanymi wyłącznie przez powłoki. Z kontrolą zadań związanych jest kilka ważnych pojęć:

zadanie pierwszoplanowe

Działa w powłoce, zajmując ją tak, że nie jest możliwe wydanie kolejnych poleceń.

zadanie w tle

Działa w powłoce, ale nie zajmuje jej, w związku z czym możliwe jest wydawanie kolejnych poleceń w tej samej powłoce.

zawieszenie

Tymczasowe wstrzymanie działania zadania pierwszoplanowego.

wznowienie

Powoduje wznowienie działania wstrzymanego wcześniej zadania.

jobs

Wbudowane polecenie `jobs` wypisuje listę zadań działających aktualnie w powłoce.

```
$ jobs
[1]- Running          emacs mojplik &
[2]+ Stopped          su
```

Liczba całkowita z lewej strony to numer zadania, a znak dodawania (+) oznacza domyślne zadanie, na które wpływ będą miały polecenia `fg` (*foreground* — pierwszy plan) i `bg` (*background* — tło).

&

Znak `&` umieszczony na końcu wiersza polecenia powoduje, że polecenie zostanie uruchomione jako zadanie w tle.

```
$ emacs mojplik &
[2] 28090
```

Powłoka odpowie, wypisując numer zadania (2) i identyfikator procesu polecenia (28090).

^Z

Naciśnięcie klawiszy ^Z w powłoce, w czasie gdy na pierwszym planie działa jakieś zadanie, spowoduje wstrzymanie tego zadania. Zadanie przestaje działać, ale jego stan jest zapamiętywany.

```
$ mojewielkiprogram
^Z
[1]+  Stopped                  mojewielkiprogram
$
```

Teraz możliwe jest wpisanie polecenia bg i odesłanie zadania w tło lub wpisanie fg i wznowienie zadania na pierwszym planie.

suspend

Wbudowane polecenie suspend, jeżeli jest to możliwe, spowoduje wstrzymanie aktualnej powłoki, tak jakby zostały naciśnięte klawisze ^Z w odniesieniu do samej powłoki. Na przykład jeżeli wpisane zostanie polecenie su, a następnie użytkownik będzie chciał wrócić do pierwotnej powłoki:

```
$ whoami
kowalski
$ su -l
Password: *****
# whoami
root
# suspend
[1]+  Stopped                  su
$ whoami
kowalski
```

bg

bg [%numer_zadania]

Wbudowane polecenie bg odsyła wstrzymane zadanie do pracy w tle. Wpisane bez żadnych argumentów polecenie bg będzie działać na ostatnio wstrzymanym zadaniu. Aby podać konkretne zadanie (z listy podawanej przez polecenie jobs), należy wpisać numer tego zadania poprzedzony znakiem procenta (%).

```
$ bg %2
```

Niektóre rodzaje zadań interaktywnych nie mogą działać w tle, na przykład ze względu na to, że oczekują na podanie danych. Jeżeli użytkownik będzie próbował odesłać takie zadanie w tło, powłoka wstrzyma je i wypisze na ekranie:

Teraz możliwe jest tylko wznowienie zadania poleceniem fg.

fg

```
fg [%numer_zadania]
```

Wbudowane polecenie fg przywraca na pierwszy plan zadanie działające w tle lub wstrzymane. Wpisane bez żadnych argumentów wybiera zadanie ostatnio wstrzymane lub odesłane w tło. Aby podać konkretne zadanie, należy wpisać numer tego zadania poprzedzony znakiem procenta (%):

```
$ fg %2
```

Zabijanie działającego polecenia

Jeżeli polecenie uruchomione w powłoce musi zostać natychmiast zatrzymane, należy nacisnąć klawisze ^C. Dla powłoki naciśnięcie tych klawiszy oznacza: „Natychmiast zakończ działanie zadania pierwszoplanowego”. W związku z tym, jeżeli wyświetlany jest bardzo duży plik (na przykład poleceniem cat) i chcielibyśmy zatrzymać ten proces, wystarczy wcisnąć klawisze ^C:

```
$ cat wielkiplik
To jest bardzo duży plik z wieloma wierszami tekstu bla bla
↳bla bla bla bla bla bla bla bla blablablabla ^C
$
```

Aby zabić zadanie działające w tle, można przywołać je na pierwszy plan poleceniem fg i nacisnąć klawisze ^C, można też zastosować polecenie kill (proszę zobaczyć podrozdział „Kontrola procesów”).

Stosowanie klawiszy ^C to nie najlepszy sposób na kończenie pracy programów. Jeżeli tylko program posiada własną procedurę zakończenia — należy ją wykorzystać (więcej informacji w ramce na następnej stronie).

Kombinacja klawiszy ^C działa jedynie w powłoce. Najprawdopodobniej nie będzie miała żadnego wpływu na okna nie będące oknami powłoki. Dodatkowo niektóre programy pisane są tak, aby „przechwytywać” tę kombinację klawiszy i ignorować ją. Przykładem takiego programu może być edytor tekstu emacs.

Jak przeżyć zabicie

Zabicie pierwszoplanowego programu za pomocą klawiszy `^C` może spowodować, że powłoka będzie działać w dziwnym trybie, może nie wypisywać na ekranie naciskanych klawiszy. Tak się dzieje, ponieważ zabicie programu nie daje mu możliwości „posprzątanía po sobie”. Jeżeli to się zdarzy:

1. Należy nacisnąć klawisze `^J`. Daje to takie same efekty, jak klawisz *Enter*, ale zadziała na pewno nawet wtedy, gdy klawisz *Enter* nie działa.
2. Należy wpisać polecenie `reset` (nawet jeżeli litery nie pojawiają się na ekranie w czasie wpisywania) i nacisnąć klawisze `^J` w celu uruchomienia tego polecenia. To powinno przywrócić powłokę do normalnego stanu.

Kończenie działania powłoki

Aby zakończyć działanie powłoki, należy wpisać polecenie `exit` lub nacisnąć klawisze `^D`⁷.

```
$ exit
```

Dostosowywanie zachowań powłoki

Aby wszystkie powłoki uruchamiane w systemie działały dokładnie w ten sam sposób, należy odpowiednio edytować pliki `.bash_profile` i `.bashrc` w swoim katalogu domowym. Te pliki uruchamiane są za każdym razem, gdy użytkownik loguje się do systemu (`~/.bash_profile`) lub otwiera powłokę (`~/.bashrc`). Można w nich ustalać zmienne i aliasy, uruchamiać programy, drukować swój horoskop i wykonywać dowolne inne operacje.

Te dwa pliki są przykładami *skryptów powłoki*: plików wykonywalnych, zawierających polecenia powłoki. Na ten temat więcej informacji można znaleźć w podrozdziale „Programowanie skryptów powłoki”.

To koniec krótkiego wprowadzenia do Linuksa i powłoki. W dalszej części zostaną omówione wymienione i opisane najbardziej przydatne polecenia służące do pracy z plikami, procesami, użytkownikami, multimediami, w sieci i inne.

⁷ Znak `Control+D` dla każdego programu odczytującego dane ze standardowego wejścia oznacza „koniec pliku”. W tym przypadku programem jest sama powłoka, która kończy swoje działanie.

Podstawowe operacje na plikach

- `ls` Wypisuje pliki z podanego katalogu.
- `cp` Kopiuje plik.
- `mv` Zmienia nazwę pliku („przesuwa” go).
- `rm` Usuwa podany plik.
- `ln` Tworzy dowiązania do plików (ich nazwy alternatywne).

Jednymi z pierwszych operacji, jakie wykonują użytkownicy w systemie Linux, jest manipulowanie plikami: kopiowanie, zmiana nazw, usuwanie i tak dalej.

ls	<code>stdin</code>	<code>stdout</code>	<code>-file</code>	<code>--opt</code>	<code>--help</code>	<code>--version</code>
-----------	--------------------	---------------------	--------------------	--------------------	---------------------	------------------------

`ls [opcje] [pliki]`

Polecenie `ls` (wymawia się je *el-es*) wypisuje atrybuty plików i katalogów. Możliwe jest wypisanie plików z aktualnego katalogu:

```
$ ls
```

z podanych katalogów:

```
$ ls katalog1 katalog2 katalog3
```

lub poszczególnych plików:

```
$ ls plik1 plik2 plik3
```

Najważniejszymi opcjami tego polecenia są opcje `-a`, `-l` i `-d`. Domyślnie polecenie `ls` ukrywa wszystkie pliki, których nazwy rozpoczynają się od kropki — zgodnie z informacjami zamieszczonymi w ramce „Pliki z kropką”. Zastosowanie opcji `-a` pozwala na wypisanie wszystkich plików.

```
$ ls
mojplik1 mojplik2
$ ls -a
.ukryty_plik mojplik1 mojplik2
```

Opcja `-l` powoduje, że polecenie podaje więcej informacji o plikach:

```
-rw-r--r--  1 kowalski uzytkownicy  149 Paz 28 2002 moje.dane
```

w których zawarte są od lewej do prawej: uprawnienia do pliku (`-rw-r--r--`), właściciel pliku (`kowalski`), grupa (`uzytkownicy`), rozmiar (`149` bajtów), data ostatniej modyfikacji (`28 października 2002 roku`) i nazwa pliku. Dokładniejsze informacje na ten temat można znaleźć w podrozdziale „Zabezpieczenia plików”.

Opcja `-d` powoduje wyświetlenie informacji o samych katalogach, pomijane są informacje o plikach znajdujących się w katalogu:

```
$ ls -ld moj katalog
drwxr-xr-x 1 kowalski kowalski 4096 wrz 29 2011 moj katalog
```

Przydatne opcje

- a Wypisuje wszystkie pliki, włącznie z tymi, których nazwy zaczynają się od kropki.
- l Dłuższy wydruk, zawierający atrybuty plików. Dodanie opcji `-h` spowoduje, że wielkości plików będą podawane w kilobajtach, megabajtach i gigabajtach, a nie w bajtach.
- F Pewne nazwy plików uzupełniane są specjalnymi symbolami, oznaczającymi typy plików. Do nazw katalogów dodawany jest znak lewego ukośnika (`/`), do plików wykonywalnych — znak gwiazdki (`*`), do dowiązań symbolicznych — znak (`@`), do potoków nazwanych — znak pionowej kreski — (`|`), a do gniazdek — znak równości (`=`). Są to tylko graficzne oznaczenia typu plików przeznaczone dla użytkowników; nie są one częścią nazw tych plików.
- i Przed nazwami plików dodawane są numery węzłów inode.
- s Przed nazwami plików dodawane są ich rozmiary, co bardzo przydaje się przy sortowaniu plików według ich wielkości.

```
$ ls -s | sort -n
```
- R Jeżeli wypisywana jest zawartość katalogu, wypisywana jest też zawartość jego podkatalogów.
- d Jeżeli wypisywana jest zawartość katalogu, sama zawartość jest pomijana, a wypisywany jest sam katalog.

cp stdin stdout -file --opt --help --version
cp [opcje] pliki (plik | katalog)

Polecenie `cp` normalnie wykonuje kopiowanie pliku:

```
$ cp plik plik2
```

albo kopiuje wiele plików do jednego katalogu:

```
$ cp plik1 plik2 plik3 plik4 katalog_docelowy
```

Za pomocą opcji `-a` możliwe jest też rekursywne kopiowanie zawartości katalogów.

Przydatne opcje

- p Kopiowana jest nie tylko zawartość pliku, ale również wszystkie związane z nim uprawnienia, znaczniki czasu, a jeżeli pozwalają na to uprawnienia użytkownika, także dane jego właściciela i grupy. Normalnie właścicielem kopiowanych plików

staje się kopiujący je użytkownik, są one oznaczane aktualnym czasem, a ich uprawnienia powstają przez połączenie wartości *umask* użytkownika z oryginalnymi uprawnieniami.

- a Hierarchia katalogów jest kopiowana rekursywnie, z zachowaniem właściwości plików i dowiązań.
- r Hierarchia katalogów jest kopiowana rekursywnie. Opcja ta nie zachowuje właściwości plików, takich jak uprawnienia czy oznaczenia czasowe. Zachowane zostają dowiązania symboliczne.
- i Tryb interaktywny. Przed usunięciem plików docelowych użytkownik zostanie zapytany o pozwolenie.
- f Wymuszenie wykonania kopiowania. Jeżeli plik docelowy już istnieje, zostanie on bezwarunkowo usunięty.

mv	stdin	stdout	-file	--opt	--help	--version
-----------	-------	--------	-------	-------	--------	-----------

mv [opcje] *źródło cel*

Polecenie mv (*move* — przenieś) pozwala na zmianę nazwy pliku:

```
$ mv plik1 plik2
```

lub przeniesienie plików i katalogów do katalogu docelowego:

```
$ mv plik1 plik2 katalog3 katalog4 katalog_docelowy
```

Przydatne opcje

- i Tryb interaktywny. Przed usunięciem plików docelowych użytkownik zostanie zapytany o pozwolenie.
- f Wymuszenie wykonania przeniesienia. Jeżeli plik docelowy już istnieje, zostanie on bezwarunkowo usunięty.

rm	stdin	stdout	-file	--opt	--help	--version
-----------	-------	--------	-------	-------	--------	-----------

rm [opcje] *pliki | katalogi*

Polecenie rm (*remove* — usuń) umożliwia usuwanie plików:

```
$ rm plik1 plik2 plik3
```

lub rekursywne usuwanie całych katalogów:

```
$ rm -r katalog1 katalog2
```

Przydatne opcje

- i Tryb interaktywny. Użytkownik jest pytany o pozwolenie na usunięcie każdego z plików.
- f Wymuszenie usunięcia. Wszystkie błędy i ostrzeżenia są ignorowane.

- r Rekursywne usuwanie katalogu i jego zawartości. Tej opcji należy używać ostrożnie, szczególnie w połączeniu z opcją -f, która może spowodować usunięcie wszystkich plików użytkownika.

ln	stdin	stdout	-file	--opt	--help	--version
-----------	-------	--------	-------	-------	--------	-----------

ln [opcje] źródło cel

Dowiązanie (link) jest odnośnikiem do innego pliku, tworzonym przez polecenie ln. Można powiedzieć, że dowiązania umożliwiają nadanie temu samemu plikowi wielu nazw, dzięki czemu może on istnieć w co najmniej dwóch lokalizacjach jednocześnie.

Istnieją dwa rodzaje dowiązań. *Dowiązanie symboliczne* odnosi się do innego pliku poprzez jego ścieżkę; podobnie działają „skrótów” w systemie Windows i „aliasy” w systemie Macintosh. Aby utworzyć dowiązanie symboliczne, należy użyć opcji -s:

```
$ ln -s mojplik dowiazanie_symboliczne
```

Jeżeli oryginalny plik zostanie usnięty, wtedy dowiązanie symboliczne będzie nieprawidłowe, wskazując na nieistniejący już plik. Z drugiej strony, *dowiązanie sztywne* jest tylko inną nazwą tego samego pliku na jednym dysku. Mówiąc bardziej technicznie — wskazuje ono na ten sam węzeł inode. Usunięcie pliku oryginalnego nie powoduje unieważnienia dowiązania sztywnego. Na rysunku 5. przedstawiono różnicę między tymi typami dowiązań. Aby utworzyć dowiązanie sztywne, należy wpisać:

```
$ ln mojplik dowiazanie_sztwyne
```

Dowiązania symboliczne mogą wskazywać pliki na innych partycjach, ponieważ są odnośnikami do ścieżki pliku. Dowiązania sztywne nie mogą wykroczyć poza granice partycji, ponieważ numer węzła inode jednej partycji nie ma żadnego znaczenia w innej. Poza tym dowiązania symboliczne mogą wskazywać na katalogi, a dowiązania sztywne — nie, chyba że utworzy je superużytkownik z wykorzystaniem opcji -d.

Przydatne opcje

- s Utworzenie dowiązania symbolicznego. Domyślnie tworzone są dowiązania sztywne.
- i Tryb interaktywny. Przed usunięciem plików docelowych użytkownik zostanie zapytany o pozwolenie.
- f Wymuszenie utworzenia dowiązania. Jeżeli plik docelowy istnieje, zostanie on bezwarunkowo usunięty.
- d Tworzenie dowiązania sztywnego do katalogu (tylko dla superużytkownika).

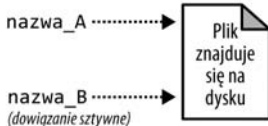
Tworzenie pliku:

```
$ touch nazwa_A
```



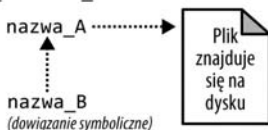
Dowiązanie sztywne:

```
$ ln nazwa_A nazwa_B
```



Dowiązanie symboliczne:

```
$ ln nazwa_A nazwa_B
```



Rysunek 5. Dowiązania sztywne i symboliczne

Za pomocą jednego z poniższych poleceń bardzo łatwo można sprawdzić, gdzie wskazuje dowiązanie symboliczne:

```
$ readlink nazwa_dowiazania
```

```
$ ls -l nazwa_dowiazania
```

A

adres
 IP, 145
 MAC, 145
Advanced Packaging Tool, 190
aktualny katalog roboczy, 18
aliasy, 31
aplikacja w ścieżce katalogu, 23
aplikacja, application, 20
argumenty wiersza poleceń, 202
atrybuty, 72
audio, 182

B

bity uprawnień, 69

C

change attribute, 72
change directory, 46
change finger, 138
change group, 68
change mode, 69
change owner, 67
change shell, 139
cudzysłowy odwrócone, 79
czas, 129
czytnik grup dyskusyjnych, 163

D

data systemu, 178
daty, 127
disk free, 109
disk usage, 65

dostęp do
 katalogu, 25
 pliku, 25
dowiązanie (link), 44
 symboliczne, 44
 sztywne, 44
drukowanie, 104
dyski, 107
dystrybucja, 6
dystrybucje Linuksa, 12

E

edycja w wierszu poleceń, 34
edytor
 emacs, 59
 domyślny, 59
 vim, 60
 wideo, 186
Eye of Gnome, 179

F

filesystem
 check, 111
 table, 110
format
 daty, 176
 GNU Zip, 95
 Zip, 95
formatowanie, 169
formatowanie dysków, 109
formaty wyjściowe, 101
FTP, File Transfer Protocol, 153
funkcja printf(), 168
funkcje powłoki bash, 27

G

GID, 132
Graficzny Interfejs Użytkownika, 13
graficzny pulpit, 13, 14
grafika, 178
grupa, 141
grupa użytkowników, 25
grupy dyskusyjne, 163
GUI, Graphical User Interface, 13

H

harmonogram zadań, 125
hasło logowania, 138
historia poleceń, 27, 35

I

identyfikator
 grupy, 132
 procesu, 118
 użytkownika, 132
informacje o
 komputerze, 143
 o użytkownikach, 134
 o zużyciu zasobów, 121
instalowanie oprogramowania, 187
instrukcja if, 196
instrukcje warunkowe, 196
interfejs
 graficzny, 6, 15
 tekstowy, 6

J

jądro (kernel), 12
języki skryptowe, 94, 203

K

katalog
 /bin/bash, 27
 /boot, 23
 /dev, 108
 /home, 19

 /lost+found, 23
 /proc, 23
 /root, 19
 /usr/bin/, 36
 główny, 17
katalogi, 17, 20
 domowe, 19
 nadrzędne, 18
 systemowe, 20
 systemu operacyjnego, 23
kategoria, category, 20
kategorie
 dokumentacji, 21
 konfiguracji, 21
 plików czasu wykonania, 22
 plików sieci WWW, 21
 programowania, 21
 programów, 21
 sprzętu, 22
 wyświetlania, 22
klawisz meta, 60
klawisze
 ^C, 39
 ^D, 40, 140
 ^Z, 38
kody
 błędów, 193
 powrotu, 193, 203
kombinacje klawiszy edytora, 35, 60
kompresowanie, 94
komunikatory, 165
komunikaty o błędach, 16
konta użytkowników, 135
kontrola procesów, 122
kontrola zadań powłoki, 36
konwersja plików binarnych, 95
kończenie
 działania powłoki, 40
 pracy, 130
kopie bezpieczeństwa, 112
korzeń, 17

L

line printer, 105
line printer remove, 105

Linux, 6
dostarczone programy, 12
jądro, kernel, 12
powłoka, shell, 13
system X, 13
list attributes, 73
litera
g, 70
o, 70
u, 70
logowanie, 13, 130
lokalizacja plików, 74

Ł

łamanie wierszy, 192
łączenie i negowanie testów, 195
łączenie poleceń, 33

M

magnetic tape, 117
makra, 94
man sshd, 150
manipulacja tekstem, 93
maski, 72
mechanizm kontroli zadań, 36

N

nagłówek polecenia, 8
nagrywarka płyt, 185
nawiasy
klamrowe, 29
kwadratowe, 8
nazwy
katalogów, 20
wieloznaczne, wildcards, 27
numer PID, 123

O

obliczenia matematyczne, 172
obsługa poczty, 158
octal dump, 53
odtworzenie audio, 182

odwrotna notacja polska, 174
okno powłoki, shell window, 15
opcja
--help, 9, 11
--version, 9
operacje
matematyczne, 173
na ciągach znaków, 173
na katalogach, 45
na plikach, 41
operator potoku (|), 33
operatory, 173

P

pakiet biurowy, 62
pakiety ICMP, 148
pakowanie plików, 94
partycje, 108
partycjonowanie dysków, 108
pętla
for, 199
until, 198
while, 198
pętle, 198
pisanie na ekranie, 167
plik
.bash_profile, 40
.bashrc, 40
.profile, 28
~/ .bash_profile, 34, 59
~/ .bashrc, 32
tar.bz2, 190
tar.gz, 190
pliki
.bz2, 95
.deb, 187, 189
.rpm, 187
.tar.gz, 187
.Z, 98
.zip, 95, 98
binarne, 52
crontab, 128
JPEG, 99
tar, 97, 98
tekstowe, 82
z kropką, 28

początek
 nagłówka, 52
 stopki, 52
 właściwego tekstu, 52
 poczta elektroniczna, 154
 podręcznik man, 52
 polecenia, 7
 argumenty, 7
 nagłówków, 8
 opcje, 7
 potoki, 7
 właściwości, 8
 polecenie
 abiword, 58, 62
 acoread, 56
 alias, 32
 aptitude, 189
 aspell, 107
 at, 126
 audacity, 184
 awk, 93
 basename, 46
 bg, 37, 38
 break, 200
 bunzip2, 97
 bzip2, 97
 cal, 175
 case, 197
 cat, 39, 48
 cd, 17, 46
 cdparanoia, 183
 cdrecord, 115
 chattr, 72
 chfn, 138
 chgrp, 68
 chmod, 69
 chown, 67
 chsh, 139
 cksum, 104
 clear, 172
 cmd, 6
 cmp, 102
 comm, 102
 compress, 98
 continue, 200
 cp, 41, 42
 crontab, 128
 cut, 86
 date, 176
 dc, 174
 df, 109
 dia, 180
 diff, 100, 101
 dirname, 46
 du, 65
 dump, 113
 echo, 10, 167
 egrep, 85
 emacs, 58
 eog, 179
 evolution, 155
 exit, 40, 203
 expr, 172, 174
 false, 195
 fdisk, 109
 fetchmail, 158
 fg, 37, 39
 fgrep, 85
 file, 66
 find, 74
 finger, 134
 firefox, 159
 floppy, 109
 free, 122
 fsck, 111
 ftp, 153
 gaim, 165
 geeqie, 179
 ghostview, 56
 gimp, 180
 gnnumeric, 58, 62
 gnuplot, 181
 grep, 53, 82, 85
 grip, 183
 groupadd, 142
 groupdel, 142
 groupmod, 142
 groups, 141
 gunzip, 96
 gv, 56
 gxine, 186
 gzip, 96
 HandBrake, 186
 head, 50

- host, 147
- hostname, 144
- id, 132
- id3tag, 184
- ifconfig, 146
- info, 11
- info bash, 27
- ip, 145
- jobs, 37
- k3b, 185
- kill, 123
- kino, 186
- ksnapshot, 179
- lame, 184
- last, 134
- less, 33, 49
- ln, 41, 44
- locate, 79
- logname, 131
- look, 106
- lpr, 105
- lprm, 105
- ls, 8, 10, 25, 41
- lsattr, 72, 73
- lynx, 159, 160
- m4, 94
- mail, 157
- mailq, 158
- man, 10
- md5sum, 103
- msg, 166
- metamail, 99
- mkdir, 47
- mkfs, 109
- mount, 110
- mplayer, 185
- mt, 117
- mutt, 155
- mv, 41, 43
- nice, 124
- nl, 48, 51
- nslookup, 148
- ntpdate, 178
- od, 53
- office, 62
- oobase, 62
- oocalc, 62
- oodraw, 62
- ooinpress, 62
- oomath, 62
- oowriter, 62
- parted, 109
- passwd, 138
- paste, 87
- pidof, 123
- ping, 148
- PowerShell, 6
- printenv, 135
- printf, 168, 170
- ps, 118
- pwd, 17, 46
- read, 193
- restore, 114
- rm, 41, 43
- rmdir, 47
- rpm, 189
- rscdiff, 101
- rsync, 116
- scp, 152
- sed, 93
- seq, 170
- sfdisk, 109
- sftp, 152, 154
- shutdown, 130
- sleep, 125
- slrn, 163
- soffice, 58
- sort, 33, 89
- spell, 107
- ssh, 150
- stat, 63
- strings, 53
- su, 38, 140
- sudo, 140
- suspend, 38
- sync, 112
- tail, 51
- talk, 166
- tar, 95
- tee, 92
- telnet, 151
- test, 194, 195
- thunderbird, 154
- top, 120

polecenie
 touch, 67
 tr, 88
 traceroute, 149
 true, 195
 tty, 167
 type, 27, 81
 umask, 71
 umount, 111
 uname, 143
 uncompress, 98
 uniq, 91
 unzip, 98
 updatedb, 80
 uptime, 119
 useradd, 136
 userdel, 137
 usermod, 137
 users, 133
 vim, 58
 w, 119
 watch, 126
 wc, 65
 wget, 161
 whereis, 81
 which, 80
 who, 33, 133
 whoami, 132
 whois, 148
 write, 166
 xargs, 77
 xcalc, 172
 xclock, 175
 xdvi, 57
 xscreensaver, 181
 xxd, 55
 yes, 170
 yum, 188
 zip, 98
 polecenie wbudowane, built-in
 command, 27
 połączenia sieciowe, 150
 pomoc dla
 GNOME, 12
 KDE, 12
 porównanie sum kontrolnych, 104
 porównywanie plików, 99
 porównywanie ze wzorcem, 93
 potoki, pipes, 7, 33
 powłoka (shell), 6, 13, 26
 bash, 13, 202
 tcsh, 74
 proces, 117, 122
 program
 abiword, 62
 acoread, 56
 audacity, 184
 cdrecord, 116
 dia, 180
 ftp, 153
 gaim, 165
 geeqie, 179
 GhostView, 56
 GIMP, 180
 gnnumeric, 63
 gnuplot, 181
 grip, 183
 gv, 57
 k3b, 185
 kill, 123
 lynx, 159
 mesg, 166
 mkisofs, 115
 mutt, 155
 sftp, 152
 slrn, 163
 soffice, 62
 ssh, 150
 string, 52
 talk, 166
 tar, 95, 98
 tty, 167
 wc, 65
 wget, 161
 who, 26, 31
 write, 166
 xdvi, 57
 xxd, 55
 programy pocztowe, 99, 155
 protokół
 FTP, 153
 SSH, 150
 przeglądanie
 plików, 48
 procesów, 117
 stron WWW, 159

przeglądarki WWW, 159
przekazanie skryptu, 202
przekierowanie
 strumienia błędów, 32
 wejścia, 32
 wyjścia, 32
przyznawanie uprawnień, 25
pulpit, 13, 14

R

remove directory, 47
root, 139
root directory, 17
rozszerzenie
 .bz2, 95
 .Z, 98
 .zip, 95, 98
RPN, Reverse Polish Notation, 174

S

secure copy, 152
secure shell, 150
serwer pocztowy, 158
serwery DNS, 148
skrótów klawiaturowe, 10, 11
skrypty powłoki, 40, 191, 201
skrypty powłoki bash, 192
sortowanie, 89
specyfikatory formatu, 169
sprawdzanie
 pisowni, 106
 ścieżki sieciowej, 149
 uprawnień, 25
standardowe wejście, 16
standardowe wyjście, 16
standardowy strumień błędów, 16
status wyjścia, 193
stdin, 8
stdout, 9
struktura katalogów, 20
suma kontrolna CRC, 104
superużytkownik, 16, 139
system plików, 17, 72, 107
system X, 13
systemy drukowania, 104
szyfrowanie połączenia, 151

Ś

ścieżka, path, 18
 aktualnego katalogu, 9
 katalogu, 21–23
 wyszukiwania, 31
ścieżki bezwzględne, 129
środowisko graficzne
 GNOME, 13
 KDE, 13
środowisko Kerberos, 151

T

tabela systemów plików, 110
tekst sformatowany, 168
testy
 ciągów znaków, 194
 numeryczne, 195
 plików, 194
tryb
 normalny, 60
 wstawiania, 60
tworzenie
 lustra, 116
 plików, 58
 skryptów powłoki, 201
typy plików, 21

U

UID, 132
umieszczenie komputera, 146
uprawnienia, 24, 70
 superużytkownika, 140
uprawnienia do
 odczytu (read), 25
 uruchamiania (execute), 25
 zapisu lub modyfikacji (write), 25
uruchamianie
 powłoki, 15
 skryptu, 202
urządzenia, 108
uzupełnianie nazw plików, 36
użytkownik, 16

W

- wartości logiczne, 193
- wejście, 16, 193
- wideo, 185
- wiersz poleceń, 6
- właściciel pliku, 25
- właściwości plików, 63
- wygaszacze ekranu, 178
- wyjście, 16, 193
- wyrażenia
 - logiczne, 173
 - regularne
 - proste, basic, 82
 - rozszerzone, extended, 82
- wyszukiwanie pliku, 75
- wyświetlanie zmiennych, 30
- wznowienie, 37

Z

- zaawansowane filtrowanie, 158
- zabicie pierwszoplanowego programu, 40
- zabijanie polecenia, 39
- zadanie, 36
 - pierwszoplanowe, 37
 - w tle, 37
- zakres, scope, 20
 - domyślny, 70
 - ścieżki katalogu, 22
- zarządzanie
 - grupami, 141
 - kontami użytkowników, 135
- zawieszenie, 37
- zdalne logowanie, 13
- zdalne przechowywanie, 112
- zdalny komputer, 153
- zmienna
 - HOME, 19, 129
 - PATH, 31
 - specjalna \$0, 202

- zmienne, 192
 - powłoki, 29
 - standardowe, 29
 - środowiskowe, 30
- znacznik
 - setgid, 70
 - setuid, 70
- znak
 - &, 37
 - cudzysłowu, 33
 - cudzysłowu odwróconego, 79
 - daszku ^, 10
 - dodawania +, 70
 - dolara \$, 7, 10, 26
 - gwiazdki *, 34, 149
 - końca wiersza, 85
 - kropki, 18
 - krzyżyka #, 10
 - myślnika -, 7, 9
 - nazwy wieloznaczonej, 34
 - nowego wiersza, 85
 - nowej linii, 85
 - odejmowania -, 70
 - pionowej kreski, 8
 - podwójnego myślnika --, 9
 - podwójnej kropki, 18
 - powrotu karetki, 85
 - równości =, 70
 - tyldy ~, 20
 - ucieczki, 19
 - ukośnika \, 34
 - ukośnika /, 17
 - zachęty #, 7, 9, 17
- znaki
 - białe, 192
 - specjalne, 34
- zużycie pamięci, 122
- zwracanie kodów powrotu, 203

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Linux. Leksykon kieszonkowy



Linux to najczęściej wybierany system operacyjny dla instalacji serwerowych. Jego wydajność, stabilność i możliwości nie mają sobie równych. Te atuty zostały dostrzeżone również przez użytkowników domowych.

System ten jest coraz częściej instalowany na ich komputerach. Graficzne środowisko pracy, takie jak GNOME czy KDE, robi świetne wrażenie na użytkownikach systemu operacyjnego głównego konkurenta. Jednak wielu z wielbicieli Linuksa wciąż najbardziej ceni konsolę – ona kryje w sobie prawdziwą siłę!

Linux. Leksykon kieszonkowy to książka dla tych, których do Linuksa zniechęca konieczność zapamiętania wielu poleceń i parametrów, oraz dla tych, którzy pracują z nim na co dzień i potrzebują podręcznej ściąg. Znajdziesz tu zestawienie poleceń najczęściej używanych w codziennej pracy. Ponadto dowiesz się, jak dopasować system do własnych potrzeb oraz jak programować skrypty powłoki. W trakcie lektury przekonasz się, że wiele zadań szybciej wykonasz w trybie tekstowym niż graficznym. Książka ta jest nieocenionym pomocnikiem każdego użytkownika systemu Linux, dlatego warto mieć ją pod ręką!

Dowiedz się:

- jak stosować wyrażenia regularne
- jak zarządzać użytkownikami
- jak błyskawicznie porównywać zawartość plików

Książka, którą musisz mieć zawsze pod ręką!

helion.pl
księgarnia
internetowa

Nr katalogowy: 11718



Helion

Sprawdź najnowsze promocje

🔗 <http://helion.pl/promocje>

Książki najchętniej czytane:

🔗 <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

🔗 <http://helion.pl/nowosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 233 98 83

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po WIĘCEJ!



KOD KORZYŚCI

ISBN 978-83-246-5602-8



Cena 24,90 zł