

Łukasz Sosna

Linux

KOMENDY I POLECENIA



Wydanie V

Helion 

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 wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/link5v>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-9061-4

Copyright © Helion S.A. 2018, 2022

Printed in Poland.

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

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

Spis treści

Wprowadzenie do systemu Linux	9
Czym jest Linux?	10
Dostępne dystrybucje. Jak wybrać odpowiednią?	11
Instalacja systemu	12
 1. Korzystanie z komputera pracującego pod kontrolą systemu Linux	 16
Środowisko pracy	16
Logowanie się do systemu	17
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	38
Tworzenie plików	39
Usuwanie plików	40
Wyświetlenie zawartości pliku	41
Zmiana dat modyfikacji plików i dostępu do nich	42

Kopiowanie plików i katalogów	44
Przenoszenie plików i katalogów oraz zmiana ich nazwy	47
Nadawanie praw dostępu do plików i katalogów	49
Zmiana hasła	54
Zmiana powłoki	55
Uzyskiwanie informacji o typie pliku	56
Zmiana właściciela i grupy pliku	57
Wyszukiwanie plików i katalogów	58
Wypisywanie liczby bajtów, słów i linii	63
Porównywanie plików lub zakresów bajtów	64
Uzyskiwanie informacji o ilości wolnego miejsca na partycjach	66
Ustalanie, ile miejsca zajmuje plik lub katalog	67
Polecenia more i less	68
Montowanie i odmontowywanie systemów plików	69
Aktualna ścieżka, pod którą pracujemy	71
Przełączanie się na konto innego użytkownika	72
Uzyskiwanie informacji o sprzęcie	72
Przeglądanie kalendarza	76
Aktualizacja daty i czasu	77
Kontrolowanie wysyłania wiadomości	82
Wysyłanie wiadomości do innego użytkownika	82
Wysyłanie wiadomości z pliku tekstowego	82
Wysyłanie komunikatów do wszystkich sieci z pliku tekstowego	83
Pokazywanie ostatnio zalogowanych użytkowników	83
Sprawdzanie, kto jest aktualnie zalogowany na naszym komputerze	85
Informacja o tym, kto jest zalogowany do systemu	85
Sprawdzanie swojej nazwy użytkownika	85
Pokazywanie lub ustawianie nazwy hosta systemowego	86
Wyświetlanie i ustalanie parametrów interfejsu sieciowego	87
Wyszukiwanie nazwy lub adresu IP zdalnego komputera	88

Sprawdzanie, czy dana domena jest już zarejestrowana	89
Sprawdzenie dostępności hosta	89
Czas, który upłynął od uruchomienia systemu	90
Generowanie pliku posiadającego zadany przez nas rozmiar	90
3. Administrowanie systemem	92
Poziom uruchomienia systemu	92
Demony usług	93
Użytkownicy	95
Grupy	97
Szukanie łańcuchów w bazie whatis	98
Procesy według zajętości pracy procesora	98
4. Tworzenie skryptów powłoki	99
Zmienne	101
Wypisywanie tekstu na ekranie użytkownika	102
Wartości logiczne	105
Polecenie test	106
Instrukcja if	110
Instrukcja case	111
Pętla while	112
Pętla until	113
Pętla for	113
Break	113
Continue	114
Argumenty pobierane z wiersza powłoki	114
Tworzenie plików w pętli	115
5. Polecenia dodatkowe	116
SSH	116
Historia poleceń użytych w powłoce	120
Wypisywanie pierwszych wierszy pliku	121
Wypisywanie ostatnich linii pliku	123

Uzyskiwanie informacji o trybie tworzenia nowych plików i katalogów	124
Wyświetlanie atrybutów plików i katalogów	124
Dodatkowe prawa dostępu do plików	125
Sprawdzanie dodatkowych prawnień do plików	126
Wyszukiwanie danych w plikach	127
Wysuwanie oraz wsuwanie tacki w napędzie CD/DVD	134
Wyświetlanie nazwy hosta	134
Archiwa TAR	135
Kompresja archiwum do formatu gzip	135
6. Midnight Commander	136
Podgląd plików tekstowych	137
Edycja pliku tekstowego	138
Edytowanie pliku tekstowego	139
Tworzenie nowego pliku	140
Otwieranie pliku	141
Wstawianie treści z innego pliku	141
Przechodzenie na początek i koniec dokumentu	142
Wstawianie i nadpisywanie	142
Cofanie	142
Kopiowanie tekstu	143
Przenoszenie tekstu	143
Zastępowanie tekstu	144
Wyszukiwanie w tekście	145
Przechodzenie do odpowiedniej linii	146
Wstawianie symbolu	147
Odświeżanie ekranu	147
Wstawianie bieżącej daty i godziny	147
Wstawianie wyniku wykonania polecenia konsoli	148
Opcje ogólne	148
Tryb zapisu	150
Kolorowanie tekstu	150

Zapisywanie ustawień	151
Kopiowanie katalogów i plików	151
Przenoszenie katalogów i plików	152
Tworzenie katalogu	153
Usuwanie plików i katalogów	153
Zaznaczanie grupy plików i katalogów	154
Lewe i prawe okno	154
Porządek sortowania	156
Widok filtrowany	157
Zmiana praw dostępu do plików i katalogów	158
Zmiana właściciela i grupy plików	159
Szybkie przechodzenie do katalogu	159
Wyszukiwanie plików	159
Zamiana paneli	161
Wyświetlanie wielkości katalogów	162
Ustawienia programu	162
Układ programu	163
Potwierdzanie	164
Zapisywanie własnych ustawień	165
Kończenie pracy programu	165
Skorowidz	166

Rozdział 4.

Tworzenie skryptów powłoki

Pisanie skryptów powłoki to programowanie odpowiednich instrukcji — programów, które usprawniają wykonywanie wielu czynności. Skrypty powłoki obsługują zmienne, instrukcje warunkowe, pętle i wiele innych przydatnych elementów.

Skrypty powłoki to po prostu zgrupowane polecenia zapisane w jednym pliku. Podobnie jak przy wpisywaniu poleceń w okienku terminala, powinieneś pamiętać o tym, jak będą one wpisywane do pliku. Każda nowa linia to nowe polecenie, więc nie można zapisywać poleceń w dwóch liniach.

Na przykład — aby wyświetlić listę zawartości swojego katalogu głównego, całe polecenie trzeba zapisać w jednej linii, ponieważ zapisanie go w dwóch liniach (lub większej ich liczbie) spowoduje błąd.

```
#!/bin/bash
vdir
/home/lukasz
```

```
[lukasz@localhost ~]$ ./skrypt
razem 44
drwx----- 5 lukasz lukasz 4096 gru 29 19:50 Desktop
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Dokumenty
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Muzyka
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Obrazy
-rw-r--r-- 1 lukasz lukasz 0 maj 24 13:04 pik.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik2.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik.txt
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Pobieranie
-rwxrwxrwx 1 lukasz lukasz 32 cze 10 13:41 skrypt
-rwxrwxrwx 1 lukasz lukasz 31 cze 10 13:41 skrypt-
drwx----- 6 lukasz lukasz 4096 maj 24 13:07 tmp
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Wideo
./skrypt: line 3: /home/lukasz: is a directory
```

Poprawnie zapisany skrypt będzie wyglądał następująco:

```
#!/bin/bash
vdir /home/lukasz
```

Wykonanie skryptu da pożądany efekt — listę zawartości katalogu głównego.

```
[lukasz@localhost ~]$ ./skrypt
razem 44
drwx----- 5 lukasz lukasz 4096 gru 29 19:50 Desktop
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Dokumenty
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Muzyka
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Obrazy
-rw-r--r-- 1 lukasz lukasz 0 maj 24 13:04 pik.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik2.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik.txt
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Pobieranie
-rwxrwxrwx 1 lukasz lukasz 31 cze 10 13:41 skrypt
-rwxrwxrwx 1 lukasz lukasz 29 cze 10 13:40 skrypt-
drwx----- 6 lukasz lukasz 4096 maj 24 13:07 tmp
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Wideo
```

Skrypty powłoki muszą zostać poprzedzone odpowiednią instrukcją odwołującą się do interpretera powłoki, której używamy.

```
#!/bin/bash
```

Dodatkowo plik taki musi mieć prawa do wykonywania, które należy nadać mu za pomocą polecenia `chmod` (opisywanego we wcześniejszej części książki).

```
[lukasz@localhost ~]$ chmod 777 skrypt
```

W celu uruchomienia skryptu należy odpowiednio go wywołać. Zapiszmy skrypt w pliku `skrypt`. W takim przypadku mamy go w katalogu głównym użytkownika i aby go wywołać, nie wystarczy wpisać jego nazwy, gdyż powłoka będzie wyszukiwała polecenia o takiej nazwie w ścieżkach wyszukiwania. Przed skryptem należy wpisać pełną ścieżkę dostępu do niego, zaczynając od znaku `/`, a gdy jesteśmy w katalogu, w którym jest umieszczony skrypt, wystarczy wpisać `./` (aktualny katalog, w którym znajduje się skrypt). W takim wypadku będziemy mieli pewność, że skrypt się uruchomi.

```
[lukasz@localhost ~]$ ./skrypt
```

Drugim sposobem uruchamiania skryptu jest użycie powłoki i przekazanie do niej skryptu w formie argumentu.

```
[lukasz@localhost ~]$ bash skrypt
razem 44
drwx----- 5 lukasz lukasz 4096 gru 29 19:50 Desktop
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Dokumenty
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Muzyka
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Obrazy
-rw-r--r-- 1 lukasz lukasz 0 maj 24 13:04 pik.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik2.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik.txt
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Pobieranie
-rwxrwxrwx 1 lukasz lukasz 31 cze 10 13:42 skrypt
-rwxrwxrwx 1 lukasz lukasz 32 cze 10 13:41 skrypt-
drwx----- 6 lukasz lukasz 4096 maj 24 13:07 tmp
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Wideo
```

Trzecim sposobem uruchomienia skryptu w aktualnej powłoce jest użycie znaku specjalnego. Dokonujemy tego za pomocą znaku ..

```
[lukasz@localhost ~]$ . skrypt
razem 44
drwx----- 5 lukasz lukasz 4096 gru 29 19:50 Desktop
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Dokumenty
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Muzyka
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Obrazy
-rw-r--r-- 1 lukasz lukasz 0 maj 24 13:04 pik.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik2.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik.txt
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Pobieranie
-rwxrwxrwx 1 lukasz lukasz 31 cze 10 13:42 skrypt
-rwxrwxrwx 1 lukasz lukasz 32 cze 10 13:41 skrypt-
drwx----- 6 lukasz lukasz 4096 maj 24 13:07 tmp
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Wideo
```

Zmienne

Zmienne to elementy, które mogą przechowywać wartości. W powłoce istnieją zmienne mogące przechowywać wartości logiczne, tekst i liczby. Nie trzeba deklarować typu zmiennej na samym początku skryptu — wystarczy podać dla niej wartość podczas wpisywania skryptu.

Zmienną definiuje przypisywana do niej wartość. Wartość do zmiennej najlepiej wpisywać w cudzysłowach (przy późniejszych manipulacjach jej wartością lub próbach użycia jej w innym miejscu skryptu cudzysłów zabezpiecza nas przed wystąpieniem błędu).

Zadeklarujmy zmienną o nazwie `zmienna` zawierającą słowo `tekst`.

```
#!/bin/bash
zmienna="tekst"
```

Jak widać, `zmienna` jest zwykłym tekstem. Przy jej deklarowaniu nie trzeba dodawać żadnych znaków specjalnych przed czy za nią.

```
#!/bin/bash
zmienna="tekst"
echo zmienna
```

Przy wyświetlaniu wartości zapisanej w zmiennej należy poprzedzić ją znakiem dolara „\$”, aby wyświetlanie zadziało, to znaczy aby wyświetliła się jej wartość, a nie nazwa zmiennej.

```
#!/bin/bash
zmienna="tekst"
echo $zmienna
```

Gdybyśmy nie dodali znaku dolara przed nazwą zmiennej w instrukcji `echo`, po wywołaniu tego skryptu zostałaby wyświetlona na ekranie wartość `tekst` zamiast wartości `zmienna`.

```
[lukasz@localhost ~]$ ./skrypt
zmienna
Skrypt bez dodania znaku dolara przed nazwą zmiennej

[lukasz@localhost ~]$ ./skrypt
tekst
Skrypt ze znakiem dolara przed nazwą zmiennej
```

Wypisywanie tekstu na ekranie użytkownika

Do wypisywania tekstu używamy kilku poleceń, spośród których najpopularniejszym jest `echo`.

W celu wypisania tekstu na ekranie użytkownika po poleceniu `echo` deklarujemy tekst, który zostanie wyświetlony po wywołaniu skryptu.

```
#!/bin/bash
echo To jest tekst
```

Po wywołaniu tego skryptu otrzymamy rezultat:

```
[lukasz@localhost ~]$ ./skrypt
To jest tekst
```

-n

Zastosowanie tego parametru nie doprowadzi do wypisania na końcu linii znaku nowej linii, dzięki czemu wszystkie informacje zostaną wypisane w jednym wierszu.

```
#!/bin/bash
echo -n To jest tekst
echo To jest tekst
```

```
[lukasz@localhost ~]$ ./skrypt
To jest tekstTo jest tekst
```

-e Za pomocą tego parametru można rozpoznać i interpretować wszystkie znaki specjalne wpisywane przez nas do skryptu. Znaki specjalne deklaruje się przez poprzedzenie ich znakiem backslasha.

```
#!/bin/bash
echo -e To jest tekst\a
```

```
[lukasz@localhost ~]$ ./skrypt
To jest tekst
```

-E Zastosowanie tego parametru powoduje nieinterpretowanie znaków specjalnych we wpisywanym tekście i pominięcie ich wykonania.

```
#!/bin/bash
echo -E To jest tekst\a
```

```
[lukasz@localhost ~]$ ./skrypt
To jest teksta
```

\a Zastosowanie tego parametru powoduje pojawienie się alarmu w postaci sygnału dźwiękowego.

```
#!/bin/bash
echo -e To jest tekst\a
```

\b Wykorzystanie tego parametru po wypisaniu tekstu pozwala przesunąć kursor o jeden znak bliżej początku tekstu.

- ```
#!/bin/bash
echo -e To jest tekst\b
```
- \c Zastosowanie tego parametru powoduje niewypisanie znaku nowego wiersza na końcu linii.
- ```
#!/bin/bash
echo -e To jest tekst\c
```
- \f Zastosowanie tego parametru powoduje wysunięcie strony i zmianę miejsca kursora w tekście.
- ```
#!/bin/bash
echo -e To jest tekst\f
```
- \n Zastosowanie tego parametru powoduje pojawienie się nowego wiersza po zakończeniu wypisywania tekstu.
- ```
#!/bin/bash
echo -e To jest tekst\n
```
- \r Zastosowanie tego parametru powoduje powrót karetki do początku linii.
- ```
#!/bin/bash
echo -e To jest tekst\r
```
- \t Zastosowanie tego parametru powoduje pojawienie się znaku tabulacji w poziomie.
- ```
#!/bin/bash
echo -e To jest tekst\t
```
- \v Zastosowanie tego parametru powoduje pojawienie się tabulacji w pionie.
- ```
#!/bin/bash
echo -e To jest tekst\v
```
- \\ Parametr ten służy do wypisania znaku backslasha.
- ```
#!/bin/bash
echo -e To jest tekst\\
```
- \ ' Zastosowanie tego parametru pozwala na wypisanie pojedynczego cudzysłowu.
- ```
#!/bin/bash
echo -e To jest tekst\'
```

\” Zastosowanie tego parametru pozwala na wypisanie podwójnego cudzysłowu.

```
#!/bin/bash
echo -e To jest tekst\”
```

\nnn Zastosowanie tego parametru pozwala na wypisanie znaku z tabeli kodów ASCII o notacji ósemkowej.

```
#!/bin/bash
echo -e To jest tekst\nnn
```

## Wartości logiczne

W powłoce — tak jak w każdym innym języku programowania — występują wartości logiczne, czyli wartości TRUE lub FALSE. W systemie wartość 0 zawsze oznacza prawdę, czyli TRUE, a jakąkolwiek inną wartość oznacza fałsz, czyli wartość FALSE.

Wszystkie programy działające w powłoce zwracają informację o tym, czy udało im się poprawnie zakończyć działanie. Wartość ta jest umieszczana w specjalnej zmiennej \$?.

```
#!/bin/bash
vdir /home/lukasz
echo $?
```

Zastosowanie tego programu powinno na końcu doprowadzić do wyświetlenia liczby określającej, czy powiodło się wyświetlenie zawartości katalogu, czy nie.

```
[lukasz@localhost ~]$./skrypt
razem 44
drwx----- 5 lukasz lukasz 4096 gru 29 19:50 Desktop
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Dokumenty
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Muzyka
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Obrazy
-rw-r--r-- 1 lukasz lukasz 0 maj 24 13:04 plik.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik2.txt
-rw-r--r-- 1 lukasz lukasz 7 maj 24 17:30 plik.txt
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Pobieranie
-rwxrwxrwx 1 lukasz lukasz 39 cze 11 18:30 skrypt
-rwxrwxrwx 1 lukasz lukasz 34 cze 11 18:20 skrypt-
drwx----- 6 lukasz lukasz 4096 maj 24 13:07 tmp
drwxrwxr-x 2 lukasz lukasz 4096 lis 24 2005 Wideo
0
```

Jak widać powyżej, katalog został wyświetlony i dlatego program zwrócił wartość TRUE, czyli liczbę 0 na końcu kodu. W przypadku niepowodzenia zwróciłby wartość 1, tak jak poniżej.

```
[lukasz@localhost ~]$./skrypt
vdir: /home/lukasz2: Nie ma takiego pliku ani katalogu
1
```

## Polecenie test

Polecenie test służy do porównywania liczb lub ciągów znaków i wpisywania do zmiennej wartości porównania.

- d Za pomocą tego parametru sprawdzamy, czy plik o podanej nazwie jest katalogiem.

```
#!/bin/bash
test -d plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- f Za pomocą tego parametru sprawdzamy, czy plik jest zwykłym plikiem, czy też z prawami do wykonywania.

```
#!/bin/bash
test -f plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- L Za pomocą tego parametru sprawdzamy, czy plik jest dowiązaniem symbolicznym.

```
#!/bin/bash
test -L plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- r Za pomocą tego parametru sprawdzamy, czy dany plik istnieje i czy można go odczytać.

```
#!/bin/bash
test -r plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- w Za pomocą tego parametru sprawdzamy, czy dany plik istnieje i czy można go zapisać.

```
#!/bin/bash
test -w plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- x Za pomocą tego parametru sprawdzamy, czy plik o danej nazwie istnieje i czy można go uruchomić.

```
#!/bin/bash
test -x plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- s Za pomocą tego parametru sprawdzamy, czy dany plik został zapisany na dysku i czy jego wartość (długość) nie jest zerowa.

```
#!/bin/bash
test -s plik.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- nt Za pomocą tego parametru sprawdzamy, czy *plik1* jest nowszy od *plik2*.

```
#!/bin/bash
test plik.txt -nt plik2.txt
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- ot Za pomocą tego parametru sprawdzamy, czy *plik1* jest starszy od *plik2*.

```
#!/bin/bash
test plik.txt -ot plik2.txt
echo $?

[lukasz@localhost ~]$./skrypt
0
```

- = Za pomocą tego parametru sprawdzamy, czy ciągi podane po jego obu stronach są identyczne.

```
#!/bin/bash
test "abc"="abc"
echo $?

[lukasz@localhost ~]$./skrypt
0
```

- != Za pomocą tego parametru sprawdzamy, czy ciągi podane po jego obu stronach nie są identyczne.

```
#!/bin/bash
test "abc"!="abc"
echo $?

[lukasz@localhost ~]$./skrypt
1
```

- z Za pomocą tego parametru sprawdzamy, czy ciąg podany za nim ma zerową długość.

```
#!/bin/bash
test -z abc
echo $?

[lukasz@localhost ~]$./skrypt
1
```

- n Za pomocą tego parametru sprawdzamy, czy ciąg podany za nim ma niezerową długość.

```
#!/bin/bash
test -n abc
echo $?

[lukasz@localhost ~]$./skrypt
0
```

- eq Za pomocą tego parametru sprawdzamy, czy wartości podane po jego obu stronach są sobie równe.

```
#!/bin/bash
test 1 -eq 1
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- ne Za pomocą tego parametru sprawdzamy, czy wartości podane po jego obu stronach nie są sobie równe.

```
#!/bin/bash
test 1 -ne 1
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- gt Za pomocą tego parametru sprawdzamy, czy pierwsza wartość jest większa od drugiej.

```
#!/bin/bash
test 2 -gt 1
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- ge Za pomocą tego parametru sprawdzamy, czy pierwsza wartość jest większa od drugiej lub równa drugiej.

```
#!/bin/bash
test 2 -ge 1
echo $?
```

```
[lukasz@localhost ~]$./skrypt
0
```

- lt Za pomocą tego parametru sprawdzamy, czy pierwsza wartość jest mniejsza od drugiej.

```
#!/bin/bash
test 2 -lt 1
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- le Za pomocą tego parametru sprawdzamy, czy pierwsza wartość jest mniejsza od drugiej lub równa drugiej.

```
#!/bin/bash
test 2 -le 1
echo $?
```

```
[lukasz@localhost ~]$./skrypt
1
```

- ! Parametr ten służy do negowania testu, bardzo często stosuje się go w instrukcjach warunkowych i w pętlach.

## Instrukcja if

Zastosowanie instrukcji pozwala sprawdzić, czy dany warunek jest spełniony, i — w zależności od tego — wykonać odpowiednie czynności.

```
if wartość
then
zrób coś
fi
Najprostszy wariant polecenia if
```

```
#!/bin/bash
if [1 = 1]
then
echo Wartosci sa rowne
fi
```

Przykład skryptu, którego użycie pozwala sprawdzić, czy wartości są równe.

```
[lukasz@localhost ~]$./skrypt
Wartosci sa rowne
```

Wykorzystanie tego warunku pozwala sprawdzić, czy wartość jest spełniona, i — w zależności od wyniku sprawdzenia — wykonać określoną operację (wartość spełniona: zrób coś, w przeciwnym wypadku: zrób coś innego).

```
if wartość
then
zrób coś
else
```

```

zrób coś innego
fi

#!/bin/bash
if [1 = 2]
then
echo Wartosci sa rowne
else
echo Wartosci sa rozne
fi

```

Przykład skryptu, którego użycie pozwala sprawdzić, czy wartości są równe, czy nie.

```

[lukasz@localhost ~]$./skrypt
Wartosci sa rozne

```

Za pomocą polecenia sprawdzamy, czy warunki są spełnione. Jeśli którykolwiek z warunków zostanie spełniony, polecenie zostanie wykonane. Jeżeli zaś żaden z warunków nie zostanie spełniony, wykonana zostanie funkcja `zrób coś innego`.

```

If wartość
then
zrób coś
elif wartość2
then
zrób coś 2
elif ...
...
else
zrób coś innego
di

```

## Instrukcja case

Zastosowanie instrukcji `case` pozwala sprawdzić, czy warunek ma odpowiednią wartość, a następnie przejść do odpowiedniego fragmentu kodu w programie.

```

Case warunek in
odpowiedz1)
zrob coś 1
;;
Esac

```

Przykład skryptu, za pomocą którego można sprawdzić, jaką wartość ma liczba.

```
#!/bin/bash
wartosc=1
case "$wartosc" in
1)
echo Liczba ma wartosc 1
;;
2)
echo Liczba ma wartosc 2
;;
Esac

[lukasz@localhost ~]$./skrypt
Liczba ma wartosc 1
```

## Pętla while

Pętla while powtarza wykonywanie określonych czynności dopóty, dopóki warunek w niej podany nie zostanie spełniony.

```
While polecenie
do
zrob coś
done
```

Oto skrypt z pętlą while, którego wykorzystanie pozwoli wyświetlić liczbę i po każdym przejściu pętli (wykonaniu fragmentu skryptu) zwiększyć ją o jeden. W przypadku gdy liczba jest równa 2, skrypt kończy działanie.

```
#!/bin/bash
i=0
while [$i -lt 2]
do
echo $i
i=`expr $i + 1`
done

[lukasz@localhost ~]$./skrypt
0
1
```

## Pętla until

Pętla `until` powtarza polecenie w niej zapisane dopóty, dopóki warunek nie zostanie spełniony.

```
Until polecenie
do
zrob coś
done
```

## Pętla for

Pętla `for` powtarza daną czynność żadaną liczbę razy (podaną w pętli).

```
For zmienna in lista
do
zrob coś
done
```

Pętla `for`, która wyświetla w kolejnych liniach wszystkie wymienione w niej owoce.

```
#!/bin/bash
for owoc in Jablko Pomarancza Cytryna
do
echo $owoc
done
```

```
[lukasz@localhost ~]$./skrypt
Jablko
Pomarancza
Cytryna
```

## Break

Polecenie `break` kończy działanie pętli, jeżeli podamy warunek potrzebny do jej zakończenia.

```
#!/bin/bash
for owoc in Jablko Pomarancza Cytryna
do
echo $owoc
if ["$owoc" = "Pomarancza"]
then
```

```
break
fi
done
```

Ten skrypt ma zakończyć działanie (za pomocą polecenia `break`) po pojawieniu się tekstu Pomarańcza.

```
[lukasz@localhost ~]$./skrypt
Jabłko
Pomarańcza
```

## Continue

Polecenie `continue` wymusza przejście do kolejnej iteracji, czyli następnego kroku.

```
#!/bin/bash
for owoc in Jabłko Pomarańcza Cytryna
do
echo $owoc
if ["$owoc" = "Pomarańcza"]
then
continue
fi
echo tekst przerywnika
done
```

W napisanym przez nas kodzie zadaniem instrukcji `continue` jest ukrycie napisu tekst przerywnika podczas przetwarzania „owocu”: Pomarańcza.

```
[lukasz@localhost ~]$./skrypt
Jabłko
tekst przerywnika
Pomarańcza
Cytryna
tekst przerywnika
```

## Argumenty pobierane z wiersza powłoki

Wszystkie napisane przez nas skrypty powłoki mogą przyjmować argumenty. Do argumentów wysłanych po uruchomieniu skryptu odwołujemy się za pomocą zmiennych `$1`, `$2`, `$3` ... `$n`.

```
#!/bin/bash
echo "Dzisiaj pogoda była $1"
```

Zastosowanie przykładowego skryptu pozwoli pobrać pierwszą wartość za jego nazwą i wstawić ją w miejsce \$1, a następnie wyświetlić ją na ekranie.

```
[lukasz@localhost ~]$./skrypt super
Dzisiaj pogoda była super
```

## Tworzenie plików w pętli

Przy zastosowaniu na przykład pętli możemy w bardzo prosty i szybki sposób utworzyć pliki w naszym katalogu. Pętlę definiujemy jako zmienną `name` zawierającą się w przedziale od 1 do 25, po czym znajduje się kropka i rozszerzenie pliku. Zbiór danych można zastąpić dowolnym innym, na przykład literami od `a` do `z`. Wewnątrz pętli dla podanego pliku wykonujemy polecenie `touch`, które w przypadku braku pliku na dysku spowoduje jego utworzenie.

```
#!/bin/bash
for name in {1..25}.txt
do
 touch $name
done
```



# Skorowidz

## A

administrowanie systemem, 92  
adres IP, 88  
aktualizacja daty i czasu, 77  
aktualna ścieżka, 71  
alias, 53  
archiwa TAR, 135

## B

bezpieczne wyłączenie, 20  
bit lepkości, 51

## C

czyszczenie terminala, 69

## D

data i czas, 77  
demony, 93, 94  
dodawanie użytkownika, 95  
domena, 89  
dostęp do pliku, 42  
dyski, 23  
dystrybucje, 11

## E

edytor tekstu, 138, 143  
opcje, 149

## F

format gzip, 135

## G

generowanie pliku, 90  
grupy, 97

## H

hasło, 54  
historia poleceń, 120

## I

informacje  
o działających usługach, 94  
o ilości wolnego miejsca, 66  
o pamięci systemowej, 75  
o sprzęcie, 72  
o trybie tworzenia plików, 124  
o typie pliku, 56  
o użytkowniku, 74  
instalacja systemu, 12, 13  
instrukcja  
case, 111  
if, 110  
interfejs sieciowy, 87

## K

kalendarz, 76  
katalogi i pliki, 21, 25  
dodatkowe prawa dostępu, 125  
edycja, 138  
generowanie, 90  
informacje o typie, 56  
kopiowanie, 44, 151

- otwieranie, 141
- podgląd, 137
- porównywanie, 64
- prawa dostępu, 49
- przenoszenie, 47, 152
- sprawdzanie uprawnień, 126
- struktura, 36
- tworzenie, 37, 39, 115, 140, 153
- usuwanie, 38, 40, 153
- wstawianie treści, 141
- wypisywanie informacji, 63
- wypisywanie linii, 121, 123
- wyszukiwanie, 58, 159
- wyszukiwanie danych, 127
- wyświetlanie atrybutów, 27, 124
- wyświetlanie wielkości
  - katalogów, 162
- wyświetlanie zawartości, 27, 41
- zmiana nazwy, 47
- zmiana dat modyfikacji, 42
- zmiana właściciela i grupy, 57
- kompresja archiwum, 135
- konto użytkownika, 72

## L

- Linux, 10
- logowanie, 17
  - tryb graficzny, 19
  - tryb tekstowy, 18

## M

- MAN, 24
- Midnight Commander, 136–165

## N

- napęd CD/DVD, 134
- nazwa
  - hosta systemowego, 86, 134
  - użytkownika, 85

## P

- pamięć systemowa, 75
- partycje, 23
- pętla
  - for, 113
  - until, 113
  - while, 112
- pliki, *Patrz* katalogi i pliki
- pobieranie argumentów, 114
- polecenie
  - adduser, 95
  - apropos, 98
  - arch, 72
  - break, 113
  - cal, 76
  - chmod, 49, 100
  - chown, 57
  - clear, 69
  - cmp, 64
  - continue, 114
  - cp, 44
  - dd, 90
  - df, 66
  - dir, 27
  - du, 67
  - echo, 102
  - file, 56
  - find, 58
  - free, 75
  - grep, 127
  - groupadd, 97
  - gzip, 135
  - head, 121
  - history, 120
  - host, 88
  - hostname, 86, 134
  - ifconfig, 87
  - last, 83
  - less, 68

## polecenie

- ls, 28
- lsattr, 126
- mc, 136
- mkdir, 37
- more, 68
- mount, 70
- mv, 48
- passwd, 54
- ping, 89
- ps, 94
- rm, 40
- rmdir, 38
- stat, 124
- su, 72
- tail, 123
- test, 106
- top, 98
- touch, 39
- umask, 124
- uname, 73
- uptime, 90
- users, 85
- vdir, 27
- w, 75
- wall, 82
- wc, 63
- who, 85
- whoami, 85
- whois, 89
- write, 82

pomoc, 24

powłoka, 55

poziom uruchomienia systemu, 92

prawa dostępu, 30, 49, 125, 158

PuTTY, 116

## S

skrypty, 99

sortowanie, 156

sprawdzenie dostępności hosta, 89

SSH, 116

system plików, 69

## Ś

ścieżka, 71

środowisko pracy, 16

## T

TAR, 135

tekst, 143–145

typ elementu, 30

## U

usługi, 93

użytkownik, 20, 95

## W

wartości logiczne, 105

widok filtrowany, 157

wyrażenia regularne, 131

wysyłanie wiadomości, 82

wyświetlanie nazwy hosta, 134

## Z

zasoby komputera, 25

zmiana

- daty dostępu, 42
- hasła, 54
- nazwy pliku, 47
- powłoki, 55
- praw dostępu, 158
- właściciela i grupy, 57, 159

zmienne, 101



# PROGRAM PARTNERSKI

— GRUPY 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  
**Helion** 

# Moc Linuksa w Twoich rękach!

- Pracuj z systemem z poziomu linii poleceń
- Sprawdź, co zrobić w przypadku awarii interfejsu graficznego
- Twórz własne skrypty powłoki i ciągi instrukcji
- Przeprowadzaj operacje na plikach oraz katalogach za pomocą menedżera plików

Wybór systemu operacyjnego, pod którym będzie pracował Twój komputer, w istocie sprowadza się do wyboru między standardowym produktem giganta z Redmond a niezawodnym, darmowym, fantastycznie elastycznym Linuksem. Ten ostatni system może być dla Ciebie niezastąpionym środowiskiem pracy, ale najpierw warto go trochę ośwoić, a przede wszystkim nauczyć się z nim porozumiewać — nie tylko z poziomu interfejsu graficznego, lecz także z poziomu linii poleceń. Ta wiedza pozwoli Ci bez trudu wydawać nawet najbardziej skomplikowane polecenia, które Twój komputer spełni w lot. Sprawdź, a sam się przekonasz.

W tej książce znajdziesz najróżniejsze komendy i polecenia pozwalające na efektywną komunikację z komputerem. Zobaczysz, jak zainstalować Linuksa i poruszać się po tym systemie oraz tworzyć własne skrypty powłoki — na przykład w celu zautomatyzowania codziennie wykonywanych operacji. Dowiesz się także, jak używać programu Midnight Commander, optymalnie zarządzać zasobami komputera i sprawnie administrować systemem. Przeglądny układ zapewni Ci łatwe wyszukiwanie określonych komend, a czytelne objaśnienia wraz z parametrami pomogą Ci właściwie je zastosować.

- Wprowadzenie do systemu Linux
- Korzystanie z komputera pracującego pod kontrolą systemu Linux
- Zarządzanie zasobami komputera
- Administrowanie systemem
- Tworzenie skryptów powłoki
- Polecenia dodatkowe
- Midnight Commander

## Poczuj się w Linuksie jak ryba w wodzie!

**Helion**

**helion.pl**

**HELION SA**  
ul. Kościuszki 1c  
44-100 Gliwice  
tel.: 32 230 98 63  
helion@helion.pl

*Sprawdź nasze szkolenia!*

**SZKOLENIA**

**AKADEMIA IT & BUSINESS**

**HELIONSZKOLENIA.PL**

**KOD KORZYŚCI**  
Sięgnij po więcej! ▶



ISBN 978-83-283-9061-4



9 788328 390614