

Paweł Krugiołka

Linux

JAK DOSTROIĆ BESTIĘ DO SWOICH POTRZEB?



ODKRYJ DOBRE STRONY **LINUXSA!**

- Idealnie od początku, czyli jak zainstalować Linuksa zgodnego z Twoimi oczekiwaniami
- Obciążenie systemu, czyli jak skompresować dane i sprawdzić, co spowalnia działanie komputera
- Sieci w Linuksie, czyli jak powinna wyglądać konfiguracja kart sieciowych i wybranych serwerów



Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek 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.

Redaktor prowadzący: Michał Mrowiec
Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

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?linuxj>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-5156-6

Copyright © Helion 2012

Printed in Poland.

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

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

Spis treści

O autorze	5
Wstęp	7
Rozdział 1. Przygotowania do zoptymalizowanej instalacji	9
Cel optymalizacji i odciążenia systemu operacyjnego Linux	10
Sprawdzenie Linuksa przed instalacją	11
Instalacja właściwa	14
Rozdział 2. Linux po pierwszym uruchomieniu	19
Poznanie struktury plików	19
Ważne pliki, wymagające kopii przed modyfikacją	21
Zarządzanie oprogramowaniem	26
Pakiety rpm i mechanizm yum	26
Pakiety deb i mechanizm apt-get	31
Mechanizm chkconfig	33
Rozdział 3. Kompresja, archiwizacja i kopie zapasowe danych	37
Cele i metody kompresji	37
Programy gzip i gunzip	38
Programy bzip2 i bunzip2	41
Archiwizacja plików	42
Kopie zapasowe	45
Rodzaje i strategie tworzenia kopii zapasowych	45
Sposoby tworzenia kopii zapasowych	47
Rozdział 4. Monitorowanie zużycia zasobów	51
Przestrzeń dyskowa i pamięć RAM	51
Kontrolowanie procesów systemowych	54
Montowanie nowych dysków twardej	58
Pamięć SWAP	65
Tworzenie pamięci SWAP na osobnej partycji	65
Tworzenie pamięci SWAP w pliku	67
Rozdział 5. Elementy konstrukcyjne i wyszukiwanie plików	69
Rodzaje elementów konstrukcyjnych	69
Wyszukiwanie plików	72

Rozdział 6. Planowanie zadań systemowych	77
Rozdział 7. Użytkownicy, grupy i uprawnienia	81
Tworzenie, modyfikacja i usuwanie użytkowników	82
Praca z grupami	89
Uprawnienia standardowe	91
Uprawnienia specjalne	95
Listy dostępu	98
Rozdział 8. Zwiększanie odporności na awarie	101
Macierze RAID	101
Konfiguracja macierzy w systemie Linux	102
Woluminy LVM	106
Konfiguracja woluminu logicznego w systemie Linux	106
Zmiana rozmiaru LVM	108
Podsumowanie mechanizmów RAID i LVM	111
Rozdział 9. Sieci w systemie Linux	113
Najczęściej używane polecenia w terminalu	113
Konfiguracja IP i DNS	117
Linux jako serwer DHCP	122
IPTABLES, czyli firewall linuksowy	123
Rozdział 10. Logi systemowe	129
Struktura i działanie mechanizmu rsyslog	129
Przykłady użycia mechanizmu rsyslog	132
Rotacja logów	134
Rozdział 11. Instalacja i optymalizacja wybranych serwerów	137
Serwer LAMP	137
Podstawowa konfiguracja serwera stron WWW	138
Tworzenie i umieszczanie na serwerze strony internetowej	140
Wirtualne hosty	141
Serwer SAMBA	143
Podstawowa konfiguracja serwera	143
Mapowanie udziału serwera SAMBA w systemie Windows XP	145
Rozbudowana konfiguracja serwera	148
Serwer FTP	150
Rozdział 12. Skrypty powłoki	153
Pobieranie danych od użytkownika	155
Instrukcje warunkowe	156
Pętle	159
Podsumowanie	162
Dodatek A Procedura resetowania hasła użytkownika root	163
Skorowidz	166

Rozdział 5.

Elementy konstrukcyjne i wyszukiwanie plików

W tym rozdziale dowiemy się, jak ułatwić sobie i skrócić codzienne czynności związane z wykorzystywaniem terminalu. Polecenia w terminalu nie muszą być wykonywane pojedynczo. W łatwy sposób można jednym wpisem utworzyć plik tekstowy, w jakim będą logi z ostatnich pięciuset linijek pliku *messages*, odnoszące się tylko i wyłącznie do serwera *dhcp*. Tworząc taki plik przy użyciu pojedynczych poleceń w terminalu, bardzo szybko zapełnilibyśmy nasz ekran w całości. Aby zobaczyć, co wpisywaliśmy na początku, musielibyśmy przewinąć go do góry (przy założeniu, że pracujemy w trybie graficznym) lub za pomocą kursorów sprawdzić uprzednio wpisywane polecenia. Dużo lepiej i przejrzystiej wszystko wygląda, gdy wiele krótkich poleceń złożymy w jedno dłuższe. Właśnie do tego służą elementy konstrukcyjne. Każdy administrator powinien je dobrze opanować. Znajomość ich jest również bardzo przydatna przy pisaniu skryptów powłoki, o których będzie mowa w jednym z późniejszych rozdziałów. W dalszej części rozdziału omówione zostaną techniki związane z wyszukiwaniem plików w Linuksie. Poznamy dwa programy występujące w każdej dystrybucji tego systemu oraz dowiemy się, jak z nich korzystać. Na koniec połączymy wyszukiwanie plików z elementami konstrukcyjnymi, żeby możliwie najbardziej dostosować system do swoich potrzeb.

Rodzaje elementów konstrukcyjnych

Pierwszymi elementami konstrukcyjnymi, jakie zostaną omówione, są znaki większości. Domyślnie w systemach linuksowych wynik każdego polecenia wyświetlany jest w oknie terminalu. Jednak w bardzo prosty sposób można to zmienić. Wyobraźmy sobie, że chcemy utworzyć plik tekstowy, w którym zapiszemy informacje o aktualnym stanie zużycia pamięci operacyjnej. Spójrzmy na rysunek 5.1.

Rysunek 5.1.
*Użycie pojedynczego
 znaku większości*

```

root@Serwer:~/Dokumenty
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@Serwer Dokumenty]# ls
[root@Serwer Dokumenty]# free -m
              total        used         free     shared    buffers     cached
Mem:           1002         315         687          0          23         137
-/+ buffers/cache:      154         848
Swap:          2035           0         2035
[root@Serwer Dokumenty]# free -m > pamiecRam.txt
[root@Serwer Dokumenty]# ls
pamiecRam.txt
[root@Serwer Dokumenty]# cat pamiecRam.txt
              total        used         free     shared    buffers     cached
Mem:           1002         315         687          0          23         137
-/+ buffers/cache:      154         848
Swap:          2035           0         2035
[root@Serwer Dokumenty]#
  
```

Analizując rysunek 5.1, widzimy, jak w bardzo prosty sposób można zapisać wynik polecenia do pliku. Najpierw weszliśmy do pustego katalogu i uruchomiliśmy program `free`. Jego wynik wyświetlił się na naszym terminalu. Następnie użyliśmy tego samego polecenia z tą różnicą, że jego wynik został zapisany do pliku. Do tego celu wykorzystany został jeden znak większości wpisany między programem a nazwą nowo utworzonego pliku. Następnie zawartość pliku wyświetliliśmy na ekranie terminalu. Jak widać, Linux w naprawdę bardzo prosty sposób pozwala przekierować wynik właściwie każdego polecenia do pliku. Jednak użycie pojedynczego znaku większości ma jedną bardzo ważną właściwość. Gdyby w przed chwilą omawianym przykładzie istniał wcześniej plik o nazwie `pamiecRam.txt`, to jego zawartość zostałaby w całości skasowana i zawierałaby jedynie wynik polecenia ostatnio przekierowanego do niego. Aby zapobiec nadpisywaniu się plików, możemy używać dwóch znaków większości. Na rysunku 5.2 przedstawiamy różnice między tymi dwoma sposobami zapisu do pliku.

Rysunek 5.2.
*Użycie podwójnego
 znaku większości*

```

root@Serwer:~/Dokumenty
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@Serwer Dokumenty]# ls
[root@Serwer Dokumenty]# echo "zapisane do pliku"
zapisane do pliku
[root@Serwer Dokumenty]# echo "zapisane do pliku" > plik.txt
[root@Serwer Dokumenty]# cat plik.txt
zapisane do pliku
[root@Serwer Dokumenty]# echo "nadpisanie pliku" > plik.txt
[root@Serwer Dokumenty]# cat plik.txt
nadpisanie pliku
[root@Serwer Dokumenty]# echo "dodanie tekstu do pliku" >> plik.txt
[root@Serwer Dokumenty]# seq 4 >> plik.txt
[root@Serwer Dokumenty]# cat plik.txt
nadpisanie pliku
dodanie tekstu do pliku
1
2
3
4
[root@Serwer Dokumenty]#
  
```

Na powyższym rysunku pokazujemy różnice między stosowaniem jednego a dwóch znaków większości. Jak widać, w przypadku wykorzystania jednego znaku większości zawartość pliku zostaje nadpisana. Natomiast używając dwóch znaków większości, nie nadpiszemy pliku, a jedynie dopiszemy coś do jego zawartości na końcu. Możemy więc w jednym pliku mieć informacje dotyczące pamięci RAM oraz zużycia dysku twardego. Dodatkowo między jednym a drugim wpisem może znaleźć się stosowny

komentarz. Wszystko to uzyskamy z pomocą kilku poleceń w terminalu. Co ciekawe, taki efekt możemy uzyskać nawet przy użyciu jednego polecenia. Doskonale zilustrowano to na rysunku 5.3.

Rysunek 5.3.
Łączenie poleceń

```

root@Serwer:~/Dokumenty
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@Serwer Dokumenty]# free > plik.txt
[root@Serwer Dokumenty]# echo " Powyżej sa informacje o pamięci ram, a poniżej o miejscu
jako zajmowanym przez obecny katalog" >> plik.txt
[root@Serwer Dokumenty]# du -h >> plik.txt
[root@Serwer Dokumenty]# cat plik.txt
total          used          free          shared    buffers         cached
Mem:           1026348      323872       702476           0         24956        140804
-/+ buffers/cache:    158112         868236
Swap:          2084852           0         2084852
Powyżej sa informacje o pamięci ram, a poniżej o miejscu zajmowanym przez obecny katalo
g
Log
12K
[root@Serwer Dokumenty]# free -m > plik2.txt ; echo " Powyżej sa informacje o pamięci
ram, a poniżej o miejscu zajmowanym przez obecny katalog" >> plik2.txt ; du -h >> plik
2.txt
[root@Serwer Dokumenty]# cat plik2.txt
total          used          free          shared    buffers         cached
Mem:           1002          316           686           0            24           137
-/+ buffers/cache:    154            847
Swap:           2035           0           2035
Powyżej sa informacje o pamięci ram, a poniżej o miejscu zajmowanym przez obecny katalo
g
Log
12K
[root@Serwer Dokumenty]#

```

Na powyższym rysunku przedstawiono ten sam efekt osiągnięty na dwa sposoby. W pierwszym sposobie został on osiągnięty za pomocą trzech poleceń, a w drugim z wykorzystaniem jednego rozbudowanego. Widać, że przy drugim sposobie kolejne polecenia z pierwszego zostały oddzielone znakiem średnika. Znak średnika pozwala łączyć polecenia w taki sposób, że każde z poleceń zostanie wykonane, niezależnie od tego, czy polecenie poprzednie wykona się prawidłowo, czy też nie. Innymi znakami, jakie mogą łączyć polecenia, są znaki ampersand (te, które znajdują się nad cyfrą 7 na klawiaturze, czyli &&). Jednak w ich przypadku kolejne polecenie wykona się tylko wtedy, gdy poprzednie zostało wykonane poprawnie. Gdy jakiegokolwiek z poleceń nie wykona się prawidłowo, kolejne w ogóle nie będą się wykonywać. Spójrzmy na rysunek 5.4.

Rysunek 5.4.
Różnice w łączeniu poleceń

```

root@serwer:~/Dokumenty
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@serwer Dokumenty]# ls
[root@serwer Dokumenty]# touch /aaa/bbb.txt ; seq 6 > /root/Dokumenty/plik.txt ; cat
/root/Dokumenty/plik.txt
touch: nie można dotknąć \'/aaa/bbb.txt\': Nie ma takiego pliku ani katalogu
1
2
3
4
5
6
[root@serwer Dokumenty]# touch /aaa/bbb.txt && seq 6 > /root/Dokumenty/plik2.txt ; cat
/root/Dokumenty/plik2.txt
touch: nie można dotknąć \'/aaa/bbb.txt\': Nie ma takiego pliku ani katalogu
cat: /root/Dokumenty/plik2.txt: Nie ma takiego pliku ani katalogu
[root@serwer Dokumenty]#

```

Jak widać na rysunku 5.4, najpierw do łączenia poleceń zostały użyte znaki średnika. Mimo że pierwsze polecenie nie wykonało się poprawnie, fakt ten nie spowodował zaprzestania wykonywania kolejnych poleceń. Dowodem tego jest wyświetlenie nieistniejącego wcześniej pliku, jaki został utworzony drugim poleceniem. Inaczej było w przypadku znaków ampersand. Tutaj po pierwszym źle wykonanym poleceniu następane

nie wykonało się w ogóle. Tak więc plik o nazwie *plik2.txt* nie mógł zostać otwarty, gdyż nie istniał. W oknie terminalu pojawił się tylko stosowny komunikat o błędzie.

Ostatni łącznik poleceń to znak podwójnej pionowej linii. W jego przypadku tylko przy błędnie wykonanym poleceniu wykonują się następne. Porównanie wszystkich trzech łączników zaprezentowano na rysunku 5.5.

```

root@serwer: ~/Dokumenty
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@serwer Dokumenty]# ls
[root@serwer Dokumenty]# ping www.kertis.pl ; seq 4 > liczby.txt ; cat liczby.txt
ping: unknown host www.kertis.pl
1
2
3
4
[root@serwer Dokumenty]# ping www.kertis.pl && seq 4 > liczby2.txt && cat liczby2.txt
ping: unknown host www.kertis.pl
[root@serwer Dokumenty]# ping www.kertis.pl || echo "Nie ma internetu" > ostrzezenie.txt ; cat ostrzezenie.txt
ping: unknown host www.kertis.pl
Nie ma internetu
[root@serwer Dokumenty]#

```

Rysunek 5.5. Porównanie trzech łączników poleceń

Na powyższym rysunku zbudowano trzy wyrażenia. Użyto w nich polecenia `ping`, jakie poznamy nieco później. Dzięki niemu możemy zdiagnozować m.in. nasze połączenie z Internetem. Każde z trzech wyrażen przedstawia działanie innych łączników poleceń. W pierwszym posłużyliśmy się średnikami. Dzięki temu każde z trzech poleceń zostało wykonane niezależnie od wyniku polecenia poprzedzającego. W drugim wyrażeniu użyte zostały znaki `&&`. Pierwsze polecenie zakończyło się niepowodzeniem. Nie mamy aktywnego połączenia z Internetem, w związku z tym strona `www.kertis.pl` nie odpowiedziała na nasze zapytanie. Poskutkowało to również tym, że kolejne polecenia, czyli utworzenie pliku *liczby2.txt* i wyświetlenie jego zawartości, nie zostały wykonane. W ostatnim, trzecim wyrażeniu pierwsze polecenie nie wykonało się poprawnie. Użycie podwójnej pionowej linii jako łącznika spowodowało, że dalsze polecenia zostały wykonane. Powstał więc plik *ostrzezenie.txt* i wyświetlona została jego zawartość informująca, że nie mamy połączenia z Internetem.

Poznaliśmy techniki budowania wydajnych poleceń w terminalu systemu Linux. Każdy przyszły administrator musi opanować do perfekcji elementy konstrukcyjne. Są one bardzo przydatne, np. podczas budowania własnych skryptów, ale tego nauczymy się w dalszej części książki. W kolejnych podrozdziałach i rozdziałach będziemy dość często łączyć pojedyncze polecenia z pomocą elementów konstrukcyjnych, aby ich używanie stało się nawykiem. Przyczyni się to do optymalizacji i skrócenia czasu pracy oraz poprawy jej wydajności. Teraz pora przejść do omówienia technik wyszukiwania plików.

Wyszukiwanie plików

Wraz z kolejnymi miesiącami pracy naszej linuksowej maszyny jej dyski twarde stają się w coraz większym stopniu zapełnione plikami, czy to przez użytkowników SAMBY, serwera FTP, czy przez nas samych. Gdy nagromadzi się bardzo dużo danych, możemy

mieć większy niż na początku problem ze znalezieniem dokładnie tego zdjęcia czy pliku pdf, jaki nas interesuje. Przy takich właśnie problemach przychodzi z pomocą kilka programów, jakie będą omawiane w tym podrozdziale. Programy te używane wraz z po-znanymi wcześniej elementami konstrukcyjnymi mogą wiele zdziałać.

Pierwszym programem, którego używa się do wyszukiwania plików w Linuksie, jest program `find`. Program ten dostępny jest w każdej dystrybucji Linuksa, a jego składnia jest naprawdę bardzo prosta. Po nazwie programu podajemy miejsce, w którym szukamy, oraz parametr i nazwę pliku, którego szukamy. Polecenie to najczęściej wykorzystywane jest z następującymi parametrami.

- name — parametr, po którym podajemy nazwę pliku, którego szukamy.
- user — używamy go, gdy szukamy plików konkretnego użytkownika.
- group — stosujemy go, gdy szukamy plików należących do konkretnej grupy.
- size — służy do szukania plików o konkretnym rozmiarze. Dodatkowo definiujemy tutaj, czy rozmiar podany przez nas jest w bajtach (c), kilobajtach (k), megabajtach (M), czy może gigabajtach (G).

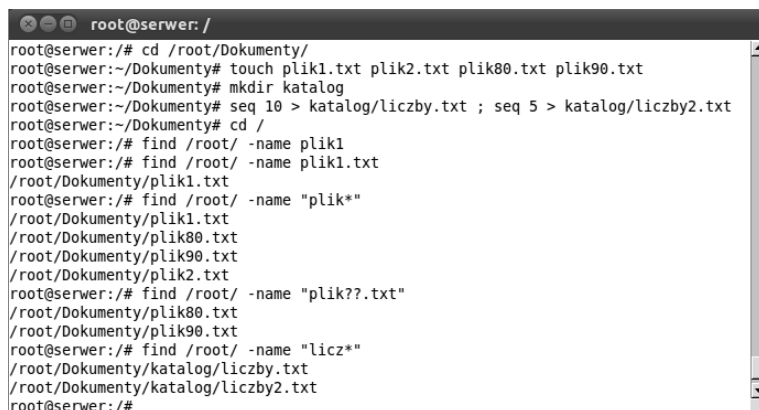
Oczywiście, w programie `find` możemy podać jednocześnie kilka parametrów, według których będą przeprowadzane poszukiwania. Wyświetlany wynik polecenia również może zależeć od tego, jakie z naszych parametrów zostały spełnione. Tutaj z pomocą przychodzi jeszcze trzy następujące parametry.

- a — w wyniku polecenia otrzymamy tylko i wyłącznie pliki spełniające wszystkie nasze kryteria.
- o — w wyniku polecenia otrzymamy pliki spełniające którekolwiek z naszych kryteriów.
- n — w wyniku polecenia otrzymamy pliki niespełniające naszych kryteriów.

Poznaliśmy już od strony teoretycznej narzędzie `find` oraz część jego możliwości. Po- ra więc przejść do praktyki. Przyjrzyjmy się rysunkowi 5.6, na którym zaprezentowa- no kilka przykładów z tym właśnie programem.

Rysunek 5.6.

Podstawowe użycie polecenia `find`



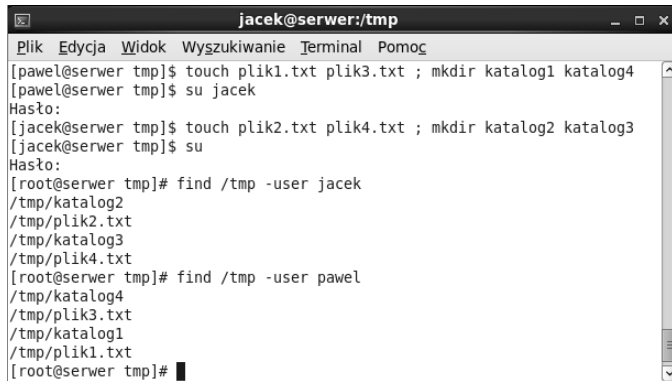
```
root@serwer: /
root@serwer:/# cd /root/Dokumenty/
root@serwer:~/Dokumenty# touch plik1.txt plik2.txt plik80.txt plik90.txt
root@serwer:~/Dokumenty# mkdir katalog
root@serwer:~/Dokumenty# seq 10 > katalog/liczby.txt ; seq 5 > katalog/liczby2.txt
root@serwer:~/Dokumenty# cd /
root@serwer:/# find /root/ -name plik1
root@serwer:/# find /root/ -name plik1.txt
/root/Dokumenty/plik1.txt
root@serwer:/# find /root/ -name "plik*"
/root/Dokumenty/plik1.txt
/root/Dokumenty/plik80.txt
/root/Dokumenty/plik90.txt
/root/Dokumenty/plik2.txt
root@serwer:/# find /root/ -name "plik??.txt"
/root/Dokumenty/plik80.txt
/root/Dokumenty/plik90.txt
root@serwer:/# find /root/ -name "licz*"
/root/Dokumenty/katalog/liczby.txt
/root/Dokumenty/katalog/liczby2.txt
root@serwer:/#
```

Na powyższym rysunku najpierw weszliśmy do katalogu `/root/Dokumenty` i utworzyliśmy trzy pliki oraz katalog, w którym umieściliśmy jeszcze dwa pliki wypełnione liczbami. Pierwsze polecenie `find` nie zwróciło żadnego wyniku. Domyślnie, jeśli po parametrze `-name` podamy część nazwy jakiegoś pliku, nie znajdziemy niczego. Musimy podać dokładną nazwę pliku lub zastosować cudzysłów, tak jak niżej na rysunku. W cudzysłów możemy dodatkowo wpisać dwa znaki. Gwiazdka zastępuje ciąg znaków, tzn. zostaną znalezione wszystkie pliki zaczynające się od znaków wpisanych przed gwiazdką, niezależnie od długości nazwy. Gwiazdka zastępuje więc ciąg dowolnych znaków w przeciwieństwie do znaku zapytania, który zastępuje tylko jeden znak. Doskonale widać to na rysunku. Co ciekawe, polecenie `find` działa rekurencyjnie. W przykładzie podaliśmy tylko katalog do poszukiwań `/root/`, a wyniki otrzymaliśmy z katalogów znajdujących się wewnątrz niego.

Przeanalizujmy teraz kolejne dwa rysunki, czyli 5.7 i 5.8, na których zademonstrowano użycie innych parametrów programu.

Rysunek 5.7.

Polecenie `find`
z parametrem `-user`



```

jacek@serwer:/tmp
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[pawel@serwer tmp]$ touch plik1.txt plik3.txt ; mkdir katalog1 katalog4
[pawel@serwer tmp]$ su jacek
Hasło:
[jacek@serwer tmp]$ touch plik2.txt plik4.txt ; mkdir katalog2 katalog3
[jacek@serwer tmp]$ su
Hasło:
[root@serwer tmp]# find /tmp -user jacek
/tmp/katalog2
/tmp/plik2.txt
/tmp/katalog3
/tmp/plik4.txt
[root@serwer tmp]# find /tmp -user pawel
/tmp/katalog4
/tmp/plik3.txt
/tmp/katalog1
/tmp/plik1.txt
[root@serwer tmp]#
  
```

Rysunek 5.8.

Polecenie `find`
z parametrem `-size`



```

root@serwer:/tmp/folder
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@serwer tmp]# mkdir folder ; cd folder
[root@serwer folder]# seq 100 > liczby.txt ; seq 10000 > liczby2.txt ; seq
1000000 > liczby3.txt ; seq 100000000 > liczby4.txt
[root@serwer folder]# du * -h
48K   liczby2.txt
6,6M  liczby3.txt
848M  liczby4.txt
4,0K  liczby.txt
[root@serwer folder]# find /tmp/folder/ -size 6M
[root@serwer folder]# find /tmp/folder/ -size +6M
/tmp/folder/liczby4.txt
/tmp/folder/liczby3.txt
[root@serwer folder]# find /tmp/folder/ -size -800M
/tmp/folder/
/tmp/folder/liczby2.txt
/tmp/folder/liczby.txt
/tmp/folder/liczby3.txt
[root@serwer folder]#
  
```

Na rysunku 5.7 pokazano użycie parametru `-user` i szukanie plików, których właścicielem jest konkretny użytkownik, natomiast na rysunku 5.8 zaprezentowano parametr `-size` i szukanie plików o konkretnych rozmiarach. Rysunek 5.7 nie wymaga komentarza, natomiast 5.8 — jak najbardziej. Należy tutaj zwrócić uwagę na znaki plus i minus użyte przed podaniem liczby określającej rozmiar pliku. Podanie rozmiaru bez znaku oznacza, że program ma znaleźć plik zajmujący dokładnie taką powierzchnię dysku,

jaka została podana. Znak plus oznacza, że powierzchnia szukanego pliku może być większa od podanej wartości, a znak minus, że powierzchnia może być mniejsza.

Przyszła pora na poznanie ostatniego już przykładu z poleceniem `find`. Przedstawiono go na rysunku 5.9.

Rysunek 5.9.

Polecenie `find`

z parametrami `-a` i `-o`



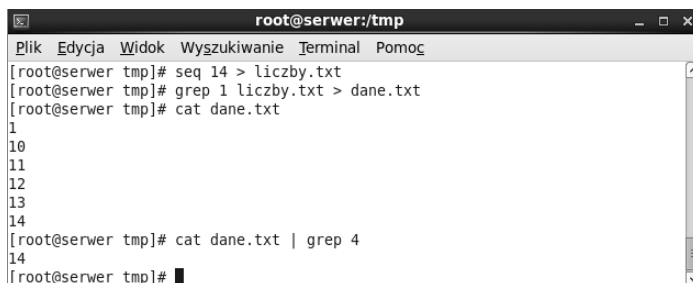
```
root@serwer:/tmp/folder
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@serwer folder]# find /tmp/folder/ -name "liczby?.txt" -a -size +6M
/tmp/folder/liczby4.txt
/tmp/folder/liczby3.txt
[root@serwer folder]# find /tmp/folder/ -name "liczby?.txt" -o -size +6M
/tmp/folder/liczby2.txt
/tmp/folder/liczby4.txt
/tmp/folder/liczby3.txt
[root@serwer folder]#
```

Na rysunku 5.9 posługujemy się plikami utworzonymi na rysunku 5.8. Mamy tutaj dwa parametry. Pierwszy wyświetla pliki spełniające wszystkie warunki, tzn. mające określoną nazwę i zajmujące więcej niż 6 MB. Drugi parametr powoduje wyświetlenie plików spełniających dowolny z dwóch warunków. Jedyny plik, jaki nie został wyświetlony w żadnym z tych przypadków, to plik o nazwie `liczby.txt`. Spowodowane jest to faktem, iż zajmuje on mniej przestrzeni niżeli 6 MB oraz ma nazwę o długości 6, a nie 7 znaków przed rozszerzeniem.

A to ostatnie polecenie niezbędne do wyszukiwania w Linuksie. Jego nazwa brzmi `grep`, a przykładowe użycie zaprezentowano na rysunku 5.10.

Rysunek 5.10.

Użycie polecenia `grep`



```
root@serwer:/tmp
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@serwer tmp]# seq 14 > liczby.txt
[root@serwer tmp]# grep 1 liczby.txt > dane.txt
[root@serwer tmp]# cat dane.txt
1
10
11
12
13
14
[root@serwer tmp]# cat dane.txt | grep 4
14
[root@serwer tmp]#
```

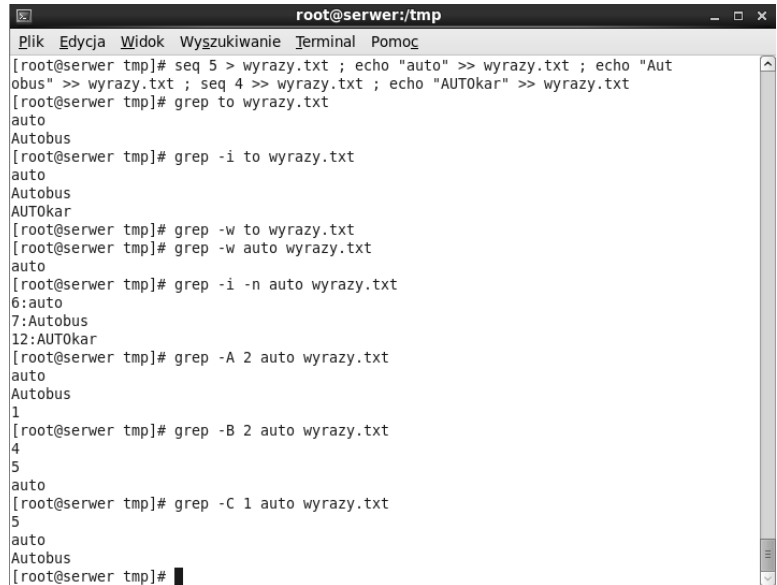
Program `grep` w głównej mierze służy do szukania danych wewnątrz plików lub do wyodrębnienia konkretnych linijek w wyniku jakiegoś polecenia. Jego użycie w tych dwóch przypadkach przedstawiono właśnie na rysunku 5.10. Najpierw utworzyliśmy plik `liczby.txt` i wypełniliśmy go liczbami od 1 do 14, a następnie przeszukaliśmy go pod kątem wszystkich wierszy zawierających liczbę 1. Po prostu po poleceniu `grep` podaliśmy szukany znak i wskazaliśmy plik, jaki chcieliśmy przeszukać. Wyniki naszych poszukiwań umieściliśmy w nowo utworzonym pliku `dane.txt`. Następnie wyświetliliśmy zawartość tego pliku na ekranie. Później użyliśmy polecenia `grep` w nieco inny sposób. Wypisaliśmy na ekranie zawartość pliku `dane.txt`, postawiliśmy pojedynczą pionową kreskę, która oznacza, że będziemy pracować na wyniku podanego przed nią polecenia. Z wyniku polecenia `cat` wypisaliśmy wszystkie wiersze zawierające cyfrę 4. Polecenia `grep` można używać naprawdę w wielu przypadkach, o czym przekonamy się w dalszych rozdziałach. Jest nieocenione przy sprawdzaniu logów czy tworzeniu skryptów. Warto wiedzieć, z jakimi parametrami występuje najczęściej. Oto one.

- R — służy do rekurencyjnego przeszukiwania plików.
- i — przy przeszukiwaniu nie zwraca uwagi na wielkość liter.
- w — umożliwi szukanie plików zawierających tylko wpisane słowo, a nie np. wyraz zawierający wpisane słowo; po wpisaniu słowa kot znajdziemy tylko i wyłącznie linijki zawierające słowo kot, a nie np. kotek czy kota.
- n — wyświetla informację, który wiersz pliku zawiera szukane przez nas słowo.
- A — wyświetla wiersz zawierający dane słowo oraz wiersz następujący po nim.
- B — wyświetla wiersz zawierający dane słowo oraz wiersz go poprzedzający.
- C — wyświetla wiersz zawierający dane słowo oraz wiersze go poprzedzające i następujące po nim.

Żeby zobaczyć, jak w praktyce wygląda posługiwanie się programem `grep` z parametrami, spójrzmy na ostatni już rysunek w tym rozdziale (rysunek 5.11).

Rysunek 5.11.

Użycie programu `grep` z parametrami



```

root@serwer:/tmp
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
[root@serwer tmp]# seq 5 > wyrazy.txt ; echo "auto" >> wyrazy.txt ; echo "Autobus" >> wyrazy.txt ; seq 4 >> wyrazy.txt ; echo "AUTOkar" >> wyrazy.txt
[root@serwer tmp]# grep to wyrazy.txt
auto
Autobus
[root@serwer tmp]# grep -i to wyrazy.txt
auto
Autobus
AUTOkar
[root@serwer tmp]# grep -w to wyrazy.txt
auto
[root@serwer tmp]# grep -w auto wyrazy.txt
auto
[root@serwer tmp]# grep -i -n auto wyrazy.txt
6:auto
7:Autobus
12:AUTOkar
[root@serwer tmp]# grep -A 2 auto wyrazy.txt
auto
Autobus
1
[root@serwer tmp]# grep -B 2 auto wyrazy.txt
4
5
auto
[root@serwer tmp]# grep -C 1 auto wyrazy.txt
5
auto
Autobus
[root@serwer tmp]#

```

Na powyższej ilustracji zaprezentowano wyniki różnego rodzaju zapytań z poleceniem `grep`.

To już wszystko w tym rozdziale. Z poznanymi tu programami będziemy stykać się w codziennej pracy z Linuxem i nieraz jeszcze użyjemy ich w tej książce. Z ich wykorzystaniem możemy np. przefiltrować pliki skrzynki pocztowych i zobaczyć, który użytkownik ma skrzynkę zajmującą największą ilość miejsca. Zastosowań jest naprawdę wiele, a wszystko zależy od naszej wyobraźni. Teraz jednak pora przejść do kolejnego rozdziału, w którym opisano planowanie zadań w Linuxie.

Skorowidz

.bash_logout, plik, 84, 85
.bashrc, plik, 84, 85, 97
.profile, plik, 84, 85

A

adduser, 83
adduser.conf, plik, 83
ampersand, znak, 71
anaconda-ks.cfg, 16, 17
Apache, 137, 138
apt-get, 31, 32
archiwizacja plików, 42, 45
at, 77
ATA, dyski, 59
atd, demon, 77
atq, 77
atrm, 77

B

bin, katalog, 20
BIOS, kolejność bootowania, 12
blkid, 64, 105
boot sequence, 12
boot, katalog, 20
bootloader, 22
bootowanie, kolejność, 12
bunzip2, 37, 41
bzip2, 37, 41
 porównanie z gzip, 41, 42

C

chage, 88, 89
 parametry, 89
chkconfig, 33, 34
 parametry, 33, 34
chmod, 92, 93, 94
chown, 91
cron, 10, 77, 78, 79
cron.allow, plik, 80
cron.deny, plik, 80
crond, demon, 78
crontab
 plik, 78, 79
 polecenie, 78, 79

D

dd, 67
deb, pakiety, 26, 31, 32
dekompresja, 37
deluser, 86
dev, katalog, 20
df, 52, 53
dhclient, 116
dhcp, pakiet, 122
DHCP, serwer, 122
dhcp.conf, plik, 122
DNS, konfiguracja, 117, 118, 119
dpkg, 31
 parametry, 32
du, 53

dyski twarde

 formatowanie, 61, 62
 montowanie, 58, 62
 odmontowanie, 62
 partycjonowanie, 59, 60, 61
 stan użycia, 52

E

echo, 153
 znaki specjalne, 153
elementy konstrukcyjne, 69, 72
etc, katalog, 20
eth0, 114
ethtool, 114

F

fdisk, 59, 60
Fedora, tworzenie pliku
 kickstart, 15, 16
find, 73, 74, 75
 parametry, 73
free, 52
fstab, plik, 24, 63
FTP, serwer, 150
 konfiguracja, 150, 151

G

getenforce, 164
getfacl, 98, 99
Gparted, 11

grep, 75
 parametry, 75, 76
 group, plik, 87, 89, 90
 groupadd, 89
 groupdel, 90
 groupmod, 90
 groups, 90
 GRUB, 22, 23
 grub.conf, 22, 23
 grupy, 89
 dodawanie, 89
 dodawanie użytkowników, 90
 modyfikacja, 90
 usuwanie, 90
 gunzip, 37, 38, 39
 gzip, 37, 38, 39
 porównanie z bzip2, 41, 42

H

home, katalog, 20
 hosty, wirtualne, 141, 142
 htop, 57
 httpd.conf, plik, 138, 139

I

ifcfg-eth0, plik, 117
 ifconfig, 113, 118
 ifdown, 115
 ifup, 115
 inittab, plik, 21
 instalacja, 14
 live cd, 11
 nadzorowana, 14
 nienadzorowana, 14, 15, 18
 zoptymalizowana, 9, 10, 11
 IP
 konfiguracja, 117, 119
 sprawdzanie adresu, 113
 IPTABLES, 123, 124
 reguły filtrowania, 123
 iptables, plik, 124
 iptables-config, plik, 124
 iptables-restore, 126
 iptables-save, 126
 iwconfig, 114

K

kickstart, 15
 tworzenie pliku, 15, 16
 kill, 55
 kompresja, 37, 38, 45

kopie zapasowe, 45
 normalna, 45
 odtwarzanie, 48, 49
 przyrostowa, 46
 różnicowa, 46
 strategię tworzenia, 46
 tworzenie, 47, 48

L

LAMP, serwer, 137
 lepki bit, 97
 lib, katalog, 20
 listy dostępu, 98, 99
 live cd, 11
 live dvd, 11
 lo, 114
 logi systemowe, 129
 rotacja, 134
 logrotate, 134
 logrotate.conf, plik, 134
 ls, 91, 92
 lsof, 57
 lvcreate, 108
 lvdisplay, 108
 LVM, woluminy, 106, 111
 konfiguracja, 106
 zmiana rozmiaru, 108
 lvresize, 108

M

macierze RAID, 101, 102, 111
 konfiguracja, 102
 mdadm, 102, 103
 mkswap, 66
 mnt, katalog, 20
 montowanie
 dysków twardych, 58, 62
 na stałe, 63
 mount, 62, 64
 Mysql, 137

N

network, plik, 119
 nmb, 143

O

oprogramowanie, zarządzanie, 26
 opt, katalog, 20

P

pamięć
 RAM, 51, 52
 SWAP, 65
 passwd, 84
 passwd, plik, 24, 85, 86, 87
 PHP, 137
 ping, 116
 pionowa linia, znak, 72
 pliki
 archiwizacja, 42, 45
 kompresja, 37, 38, 45
 struktura, 19, 21
 wyszukiwanie, 72, 73
 proc, katalog, 20
 procesy systemowe, 54
 profile, plik, 96
 przestrzeń dyskowa, 51
 ps, 54, 55
 aux, parametry, 54
 pvcreate, 106
 pvdisplay, 106

R

RAID, macierze, 101, 102, 111
 konfiguracja, 102
 RAM, pamięć, 51, 52
 reinstalacja, 21
 repo, pliki, 29
 resize2fs, 110
 resolv.conf, plik, 118
 root
 katalog, 20
 resetowanie hasła, 163,
 164, 165
 rpm, pakiety, 26, 27
 instalacja, 27
 parametry, 27
 rsyslog, 129, 130
 przykłady użycia, 132
 rsyslog.conf, plik, 130
 kategorie komunikatów, 130
 priorytety komunikatów, 131

S

SAMBA, serwer, 143
 konfiguracja, 143, 144, 145,
 148
 mapowanie udziału
 w Windows XP, 145,
 146, 147
 utworzenie użytkownika, 149

SATA, dyski, 59
 sbin, katalog, 20
 SELinux, 164
 setenforce, 164
 setfacl, 98
 sgid, 96
 shadow, plik, 87, 88
 sieci, 113

- restart interfejsów, 118
- sprawdzenie adresu IP, 113
- włączenie interfejsu, 115
- wyłączenie interfejsu, 115

 skrypty, 10, 153

- case, 158, 159
- elif, 158
- else, 158
- for, 161
- if, 157, 158
- instrukcje warunkowe, 156, 157, 158
- pętle, 159, 160, 161
- pobieranie danych od użytkownika, 155
- read, 155
- until, 160
- while, 159

 smb, 143
 smb.conf, plik, 143
 smbpasswd, 149
 snapshot, plik, 47
 sources.list, 32
 struktura plików, 19, 21
 suid, 95
 SWAP, pamięć, 65

- tworzenie, 65, 66, 67, 68

 swapoff, 66
 swapon, 66
 sys, katalog, 20
 system-config-firewall, 127
 system-config-kickstart, 15
 system-config-network, 119, 120

Ś

średnik, znak, 71

T

tail, 133
 tar, 39, 42, 43, 44, 45, 47
 tmp, katalog, 20
 top, 55, 56
 Total Commander, 151

- połączenie FTP, 152

 traceroute, 117
 tryb ratunkowy, 21
 tryby uruchamiania, 21, 22

U

Ubuntu

- instalacja, 13
- pobieranie, 11, 12
- testowanie, 13

 umask, 96, 97
 umount, 62
 uprawnienia

- przydzielanie, 92
- specjalne, 95, 97
- standardowe, 91

 useradd, 83, 84, 86

- parametry, 83, 84

 userdel, 86

- parametry, 86

 usermod, 85, 90
 usługi, 10

- sprawdzenie stanu, 34

 UUID, 64

- odczytanie, 105

 użytkownicy, 81

- dodawanie do grup, 90
- hasło, 84
- modyfikacja, 82, 85, 86

reguły tworzenia, 82
 tworzenie, 82, 83, 84
 usuwanie, 82, 86

V

var, katalog, 20
 vgcreate, 107, 108
 vgdisplay, 108
 vsftpd, pakiet, 150
 vsftpd.conf, plik, 150

W

większości, znaki, 69, 70
 wirtualne hosty, 141, 142
 woluminy LVM, 106, 111

- konfiguracja, 106
- zmiana rozmiaru, 108

 WWW, serwer, 138

- tworzenie strony internetowej, 140

 wyszukiwanie plików, 72, 73

Y

yum, 26, 28, 29, 30

- parametry, 29

 yum.conf, 28

Z

zadania systemowe,

- planowanie, 77

 zużycie zasobów, 51

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>

Linux

JAK DOSTROIĆ BESTIĘ DO SWOICH POTRZEB?

Od bardzo długiego już czasu Linux — król niekomercyjnych systemów operacyjnych — jest tak samo łatwy w obsłudze jak jego największy komercyjny rywal. Jednak w umysłach wielu osób wciąż pokutuje przekonanie, że Linux jest systemem niezwykle skomplikowanym, wymagającym zgoła lat nauki i w związku z tym niewartym ich uwagi. Pora to zmienić. Autor tej książki, na co dzień zawodowo pracujący z Linuxem, podpowie Ci, jak zmusić system, by spełniał wszystkie Twoje życzenia... i odpadywał następne!

W tej publikacji znajdziesz porady dotyczące tego, jak skonfigurować Linux na etapie instalacji, co zrobić, gdy już go odpalisz, i jak się zachowywać, by system działał bezawaryjnie. Dowiesz się, jak kompresować pliki, sprawdzać procent zużycia zasobów, nadawać uprawnienia użytkownikom czy grupom, wyszukiwać pliki i planować zadania systemowe. Ponadto zorientujesz się, jak bezpiecznie używać tego systemu, instalować serwery oraz wykorzystywać skrypty powłoki. Spróbuj, a zrozumiesz, dlaczego warto wybrać Linuksa!

- Przygotowanie do zoptymalizowanej instalacji
- Linux po pierwszym uruchomieniu
- Kompresja, archiwizacja i kopie zapasowe danych
- Monitorowanie zużycia zasobów
- Elementy konstrukcyjne i wyszukiwanie plików
- Planowanie zadań systemowych
- Użytkownicy, grupy i uprawnienia
- Zwiększanie odporności na awarie
- Sieci w systemie Linux i logi systemowe
- Instalacja i optymalizacja wybranych serwerów
- Skrypty powłoki
- Procedura resetowania hasła użytkownika root

Dzięki tej książce Linux będzie Ci służył!

Paweł Krugiołka — absolwent kierunku informatyka i ekonometria. Na co dzień pracuje jako administrator sieci w dużej korporacji oraz doradca mniejszym firmom. Do jego zadań należy zarządzanie sieciami komputerowymi. Jest certyfikowanym administratorem systemów Linux Red Hat. Współpracuje z firmami prowadzącymi kursy z tematyki sieci komputerowych i systemów operacyjnych. Informatyka jest jego hobby od lat, poza tym interesuje się wschodnimi sztukami walki i szkoleniem psów.

helion.pl
księgarnia
internetowa

Nr katalogowy: 9033



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900



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.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-5156-6



9 788324 651566

Cena: 31,00 zł

Informatyka w najlepszym wydaniu