Paweł Krugiołka

Linux

JAK DOSTROIĆ BESTIĘ DO SWOICH POTRZEB?

ODKRYJ DOBRE STRONY LINUKSA!

- 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

Helion

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

Printed in Poland.

Spis treści

Rozdział 5.

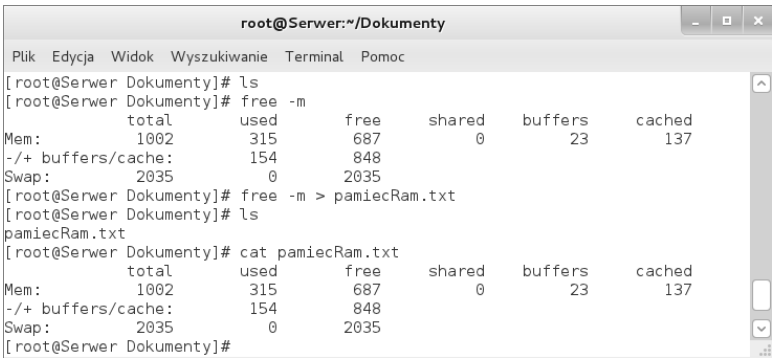
Elementy konstrukcyjne i wyszukiwanie plików

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 przejrzyściej 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

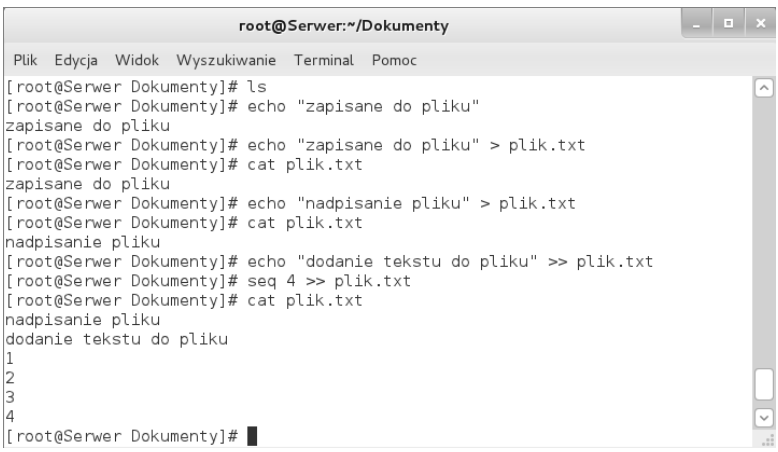
Pierwszymi elementami konstrukcyjnymi, jakie zostaną omówione, są znaki większości. Domyślnie w systemach linuksowych wynik każdego polecenia wyświetlany jest w oknie terminalu. Jednak w bardzo prosty sposób można to zmienić. Wyobraźmy sobie, że chcemy utworzyć plik tekstowy, w którym zapiszemy informacje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

Rysunek 5.1.
Użycie pojedynczego znaku większości



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 zawierałby 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.

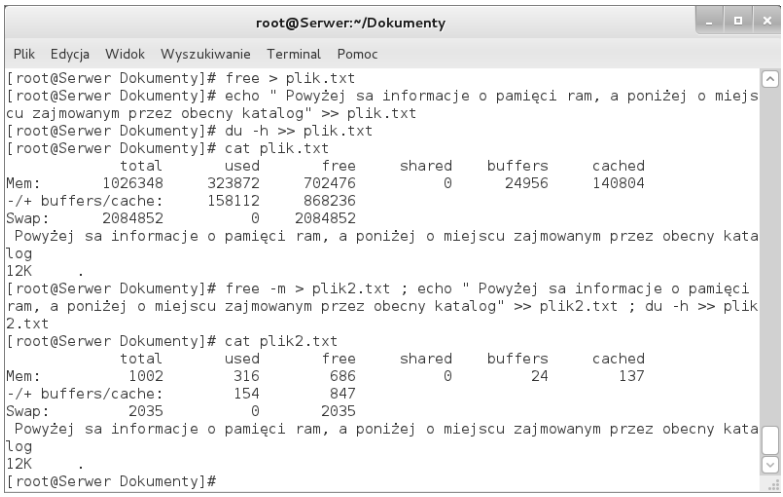
Rysunek 5.2.
Użycie podwójnego znaku większości



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

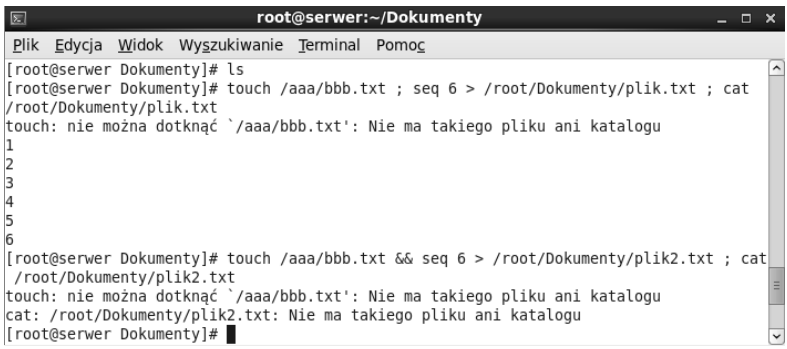
komentarz. Wszystko to uzyskamy z pomocą kilku poleceń w terminalu. Co ciekawe, taki efekt możemy uzyskać nawet przy użyciu jednego polecenia. Doskonale zilustrowano to na rysunku 5.3.

Rysunek 5.3. *Łączenie poleceń*



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 wykonanie poprawnie. Gdy jakiekolwiek z poleceń nie wykona się prawidłowo, kolejne w ogóle nie będą się wykonywać. Spójrzmy na rysunek 5.4.

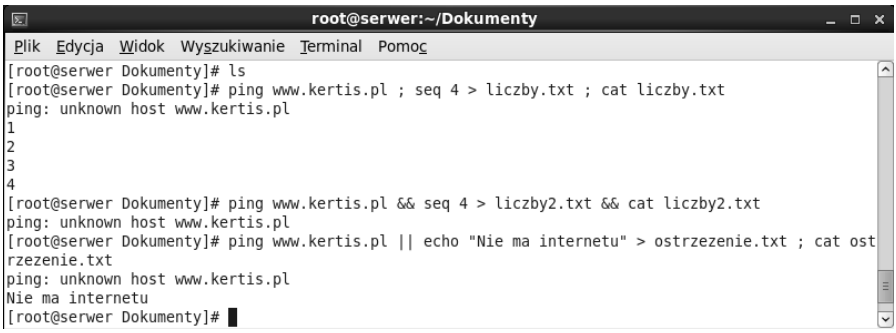
Rysunek 5.4. *Różnice w łączeniu poleceń*



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 następne

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.



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 wyrażeń 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 poznanymi 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 przychodzą jeszcze trzy następujące parametry.

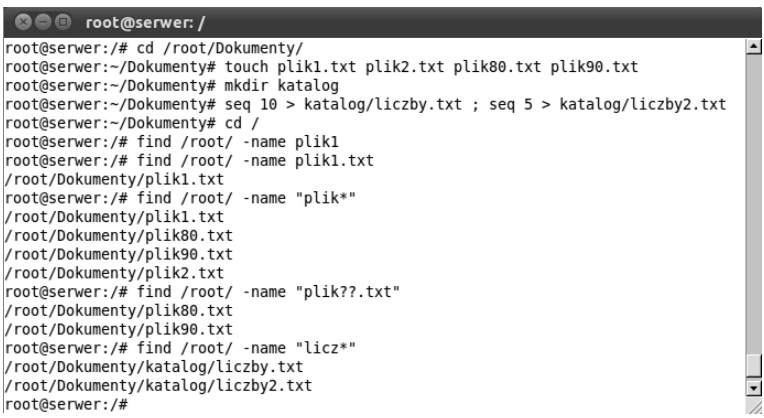
`-a` — w wyniku polecenia otrzymamy tylko i wyłącznie pliki spełniające wszystkie nasze kryteria.

`-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. Pora więc przejść do praktyki. Przyjrzyjmy się rysunkowi 5.6, na którym zaprezentowano kilka przykładów z tym właśnie programem.

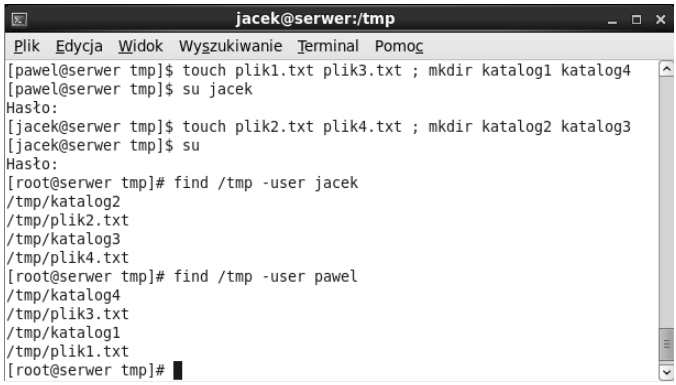
Rysunek 5.6. *Podstawowe użycie polecenia find*



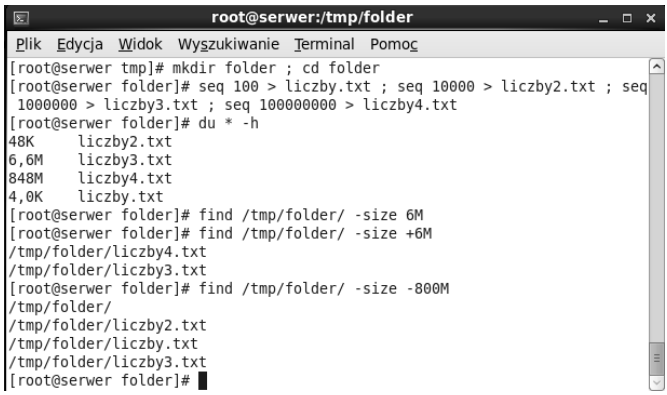
Na powyższym rysunku najpierw weszliśmy do katalogu */root/Dokumenty* i utworzyliśmy trzy pliki oraz katalog, w którym umieściliśmy jeszcze dwa pliki wypełnione liczbami. Pierwsze polecenie `find` nie zwróciło żadnego wyniku. Domyślnie, jeśli po parametrze `-name` podamy część nazwy jakiegoś pliku, nie 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*



Rysunek 5.8. *Polecenie find z parametrem -size*



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.

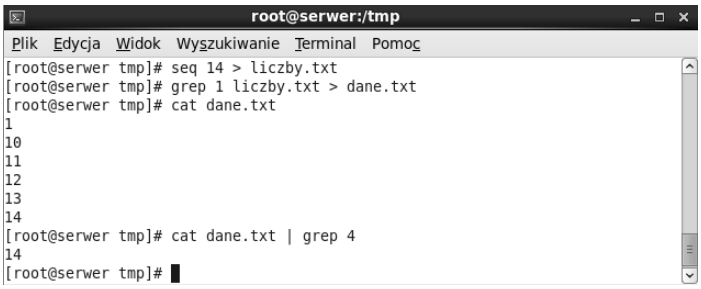
Przyszła pora na poznanie ostatniego już przykładu z poleceniem `find`. Przedstawiono go na rysunku 5.9.

Rysunek 5.9. *Polecenie find z parametrami -a i -o*

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*



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` — umożliwia 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.

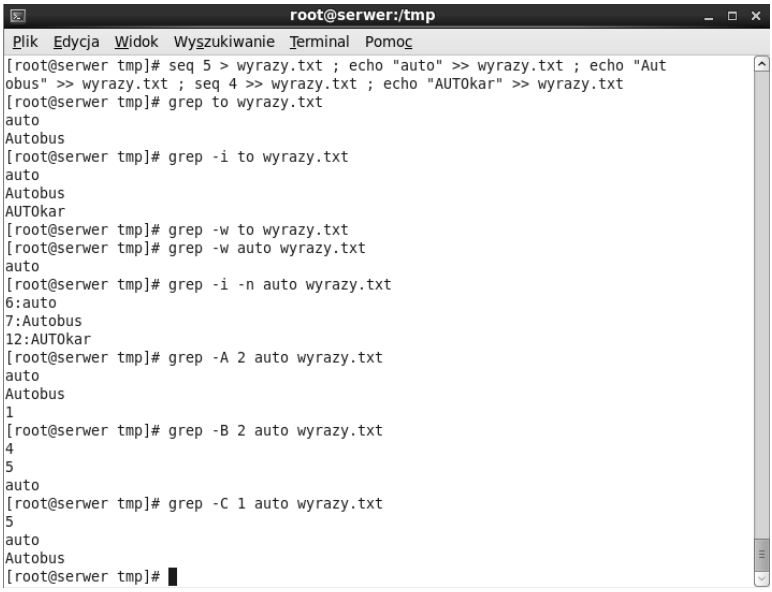
`-A` — wyświetla wiersz zawierający dane słowo oraz wiersz następujący po nim.

`-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*



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 Linuksem i nieraz jeszcze użyjemy ich w tej książce. Z ich wykorzystaniem możemy np. przefiltrować pliki skrzynek pocztowych i zobaczyć, który użytkownik ma skrzynkę zajmującą najwięcej miejsca. Zastosowań jest naprawdę wiele, a wszystko zależy od naszej wyobraźni. Teraz jednak pora przejść do kolejnego rozdziału, w którym opisano planowanie zadań w Linuksie.

Skorowidz

H

I

K

L

M

N

O

P

R

S

Ś

T

U

V

W

Y

Z

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 więksi komercyjni rywale. 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 Linuksem, podpowie Ci, jak zmusić system, by spełniał wszystkie Twoje życzenia... i odgadywał 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 doradza 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 SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-5156-6

Cena: 31,00 zł

Informatyka w najlepszym wydaniu