Debian Linux. System operacyjny dla każdego. Pierwsze starcie

Autor: Sylwester Zdanowski
ISBN: 978-83-246-2163-7
Format: 158x235, stron: 136
Zawiera CD-ROM

Przestań płacić za programy – przestaw się na Linuksa!

- Co to jest Debian Linux i czy warto z niego korzystać?
- Jak uruchomić system i używać podstawowych narzędzi?
- Na czym polega przewaga Linuksa nad produktami konkurencji?

O systemie Linux słyszał chyba każdy użytkownik komputera, a jednak nie każdy z niego korzysta – mimo dobrych opinii w środowisku informatyków. Ten zastanawiający fakt można zapewne wytłumaczyć niechęcią do oswajania się z nowym systemem, brakiem bliższych informacji na temat zalet Linuksa, różnorodności jego dystrybucji i możliwości dodatkowego oprogramowania. Dziś system ten – zaopatrzony w przyjazny interfejs użytkownika i wszelkie pomoce dodatkowe – staje się jedynym logicznym wyborem dla wszystkich, którzy cenią sobie bezpieczeństwo oraz niezawodność i nie lubią przepłacać!

Debian to jedna z trzech podstawowych dystrybucji Linuksa, którą zainstalować można na każdym – zarówno najstarszym, jak i najnowszym – komputerze. W książce „Debian Linux. Pierwsze starcie" zawarto wszelkie informacje dotyczące tej właśnie wersji systemu. Dowiesz się z niej, jak zainstalować i uruchomić system, poznasz tajniki konfiguracji oraz nauczysz się korzystać z oprogramowania – od obsługi programów biurowych, po administrowanie systemem oraz siecią. W razie problemów będziesz mógł zajrzeć do rozdziałów traktujących o zaawansowanych ustawieniach, możliwych modyfikacjach oraz dodatkowym oprogramowaniu. Przeczytasz także o zasadach bezpieczeństwa podczas pracy przy komputerze. Krótko mówiąc, znajdziesz tu wszystko, by od zaraz zacząć swoją przygodę z Linuksem!

- Instalacja systemu
- Podstawowe czynności
- Narzędzia biurowe – OpenOffice, pakiet KDE, KOffice
- Pliki PDF i drukowanie
- Nagrywanie płyt
- Administracja systemem
- Administracja siecią
- Graficzne narzędzia administratora
- Narzędzia programisty
- Jądro systemu
- SAMBA, serwer LAMP, NAGIOS, BIND
- Zabezpieczanie systemu
- Nazwy programów w systemie Linux
- Najczęściej używane polecenia

Poznaj dobry i bezpłatny system – Debian Linux!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 4.

Narzędzia informatyka

W tym rozdziale dowiesz się:

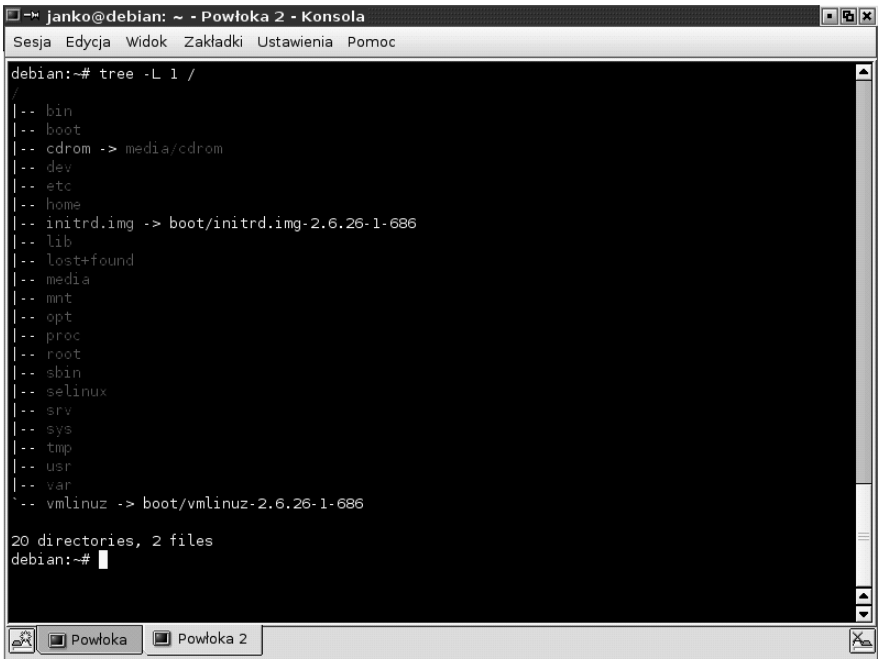
- Jak administrować systemem.
- Jak administrować siecią.
- Jak wykorzystać narzędzia graficzne administratora.
- Jakie narzędzia programisty są dostępne.

Administracja systemem

Posiadając już sprawny, funkcjonalny system Linux, można nauczyć się nim zarządzać. Wśród podstawowych czynności administracyjnych znajdują się: dodawanie i usuwanie użytkowników, zarządzanie pakietami, monitorowanie procesów, tworzenie kopii zapasowych czy konfiguracja podstawowych ustawień systemu, zarządzanie plikami i wiele innych czynności. Większość z nich można wykonać za pomocą narzędzi graficznych. Jednak w dalszym ciągu po zaznajomieniu się z linią poleceń jest ona najwygodniejsza do wykonywania wielu czynności. Ponadto umiejętność posługiwania się linią poleceń jest przydatna w przypadku problemów z systemem. Niezależnie, co spowodowało awarię systemu, problem z pojedynczym programem czy błąd w konfiguracji, linia poleceń daje możliwość naprawienia problemu bez ponownej instalacji systemu. Dlatego też narzędzia graficzne zostaną przedstawione w drugiej kolejności.

Hierarchia plików

Wykonywanie jakichkolwiek zadań administratora ułatwia znajomość hierarchii plików systemu. Podstawowy podział na katalogi widoczny jest na rysunku 4.1.



Rysunek 4.1. *Podstawa hierarchii plików*

Pierwszy katalog, *bin*, zawiera polecenia dostępne zarówno dla administratora, jak i zwykłych użytkowników. Polecenia te są również dostępne w trybie pojedynczego użytkownika. Tryb ten jest wykorzystywany w przypadku kłopotów z montowaniem dysków przy starcie systemu.

Drugi katalog, *boot*, zawiera informacje potrzebne do uruchomienia systemu. Są one wykorzystywane jeszcze przed załadowaniem jądra systemu.

Trzeci katalog, *dev*, zawiera specjalne pliki odnoszące się do zamontowanych urządzeń. Po zamontowaniu takiego urządzenia do innego katalogu możliwe jest wykonywanie na nim dozwolonych operacji, jak odczyt czy zapis plików.

Czwarty katalog, *etc*, zawiera pliki konfiguracyjne systemu oraz wszystkich programów instalowanych za pomocą pakietów. Programy instalowane ze źródeł często przechowują pliki konfiguracyjne we własnych katalogach.

Piąty katalog, *home*, przechowuje katalogi domowe oraz pliki wszystkich użytkowników. Wyjątkiem są pliki administratora, przechowywane w oddzielnym katalogu.

Szósty katalog, *lib*, przechowuje biblioteki wymagane przez polecenia znajdujące się w katalogach */bin* i */sbin*.

Siódmy katalog, *lost+found*, jest wykorzystywany w przypadku nieprawidłowego zamknięcia systemu powodującego błędy w systemie plików. Przy uruchamianiu systemu program *fsck* spróbuje odzyskać ewentualne uszkodzone pliki, czego rezultat zostanie zapisany właśnie do tego katalogu.

Ósmy katalog, *media*, zawiera podkatalogi będące miejscem montowania urządzeń takich jak dyski twarde, CD-ROM, dyskietki.

Dziewiąty katalog, *opt*, jest przeznaczony dla dodatkowych aplikacji. Można w nim dla przykładu zainstalować przeglądarkę internetową.

Dziesiąty katalog, *proc*, nie zawiera plików jako takich, jego zawartość stanowią liczne informacje na temat pracy systemu. Przykładem są informacje dotyczące procesora w pliku *cpuinfo*.

Jedenasty katalog, *root*, jest katalogiem domowym administratora systemu.

Dwunasty katalog, *sbin*, zawiera narzędzia przeznaczone do administracji systemem. Do programów tych dostęp ma tylko i wyłącznie administrator systemu.

Trzynasty katalog, *selinux*, jest wynikiem wprowadzenia w Debianie obsługi SELinux. Pozwala on na zwiększenie bezpieczeństwa systemu przez rozszerzenie kontroli dostępu.

Czternasty katalog, *srv*, jest również katalogiem relatywnie nowym w systemie Debian. Przeznaczony jest do przechowywania danych związanych z oferowanymi przez system usługami.

Piętnasty katalog, *sys*, zawiera pliki związane z pracą kernela i konfiguracją modułów oraz informacje o urządzeniach blokowych i znakowych.

Szesnasty katalog, *tmp*, zawiera pliki tymczasowe potrzebne w bieżącej pracy programów. Nie należy samemu usuwać żadnych plików z tego katalogu. System sam zajmuje się jego czyszczeniem, gdy pliki nie są potrzebne.

Siedemnasty katalog, *usr*, jest najważniejszy i największy. W nim można znaleźć dokumenty i programy dostępne dla użytkowników.

Osiemnasty katalog, *var*, zawiera logi programów, pliki tymczasowe, zmienne lokalne.

Już w pierwszym poziomie znajduje się 18 ważnych katalogów. Każdy kolejny poziom pomnaża tę liczbę. Najczęściej wykorzystywany jednak jest katalog *etc* zawierający konfigurację niemal wszystkich programów znajdujących się w systemie.

Zarządzanie pakietami

Czynnością, której nie można uniknąć w dłuższym okresie, jest instalowanie i usuwanie oprogramowania. W poprzednich działach do instalacji służyło polecenie `aptitude`. Program ten powstał na bazie narzędzi `apt-get` i `dselect`. Ważne jest, aby konsekwentnie używać tego samego programu. Ma to duże znaczenie, gdyż `aptitude` potrafi usuwać nie tylko wskazany pakiet, ale również pakiety z nim powiązane. Natomiast `apt-get` pozostawi pakiety zainstalowane ze względu na zależności. Mieszanie tych programów może doprowadzić do sytuacji, w której usunięcie zainstalowanej gry za pomocą `aptitude` spowoduje usunięcie środowiska graficznego.

Niezależnie od stosowanego narzędzia korzysta ono z tej samej podstawy. Najważniejszy jest plik */etc/apt/source.list*. Był on wykorzystywany w poprzednich rozdziałach do dodawania źródeł oprogramowania. Każda linijka tego pliku odnosi się do innego serwera, zawiera również informacje o rodzaju archiwum. Możliwe są dwa rodzaje źródeł: *deb* zawierający gotowe do instalacji pakiety lub *src-deb* z kodem źródłowym. Następna część zawiera adres serwera, może on korzystać z protokołów HTTP, FTP lub SSH. Mając do dyspozycji kilka serwerów lustrzanych, najlepszy można wybrać za pomocą programu *netselect*. W konsoli należy wpisać następujące polecenie:

```
netselect -vv ftp.debian.org ftp.pl.debian.org
```

Nazwa programu, po której dodany jest parametr `-vv`, sprawia, iż program staje się bardziej gadatliwy (ang. *verbose*). Przy użyciu pojedynczej litery `v` otrzymamy mniej informacji, parametr ten jest dostępny w wielu poleceniach. Wynikiem powyższej komendy będzie informacja, który z podanych serwerów jest szybszy. Dla ułatwienia wpisy zostaną wyświetlone w takiej właśnie kolejności, od najszybszego.

Można również korzystać z pakietów znajdujących się na płytach. W przypadku systemu testowego jest to mało użyteczne, ze względu na ilość aktualizacji. W razie takiej potrzeby płytę można dodać za pomocą polecenia `apt-cdrom add`. Po jego wywołaniu zawartość płyty zostanie odczytana i dodana do listy dostępnych pakietów.

Posiadane pakiety można instalować i usuwać na wiele sposobów. Jednym z nich jest wywołanie programu `aptitude` bez dodawania parametrów do polecenia. W górnej części okna programu widoczne jest menu, do którego można przejść, wciskając *Ctrl+T* (ta sama kombinacja klawiszy służy do opuszczania menu). Przed instalacją pakietów należy wybrać z menu: *Akcje/Uaktualnij listę pakietów*. Dzięki temu system będzie posiadał listę aktualnie znajdujących się na serwerach plików. Brak aktualizacji przed instalacją może spowodować błąd, jeżeli uległa zmianie nazwa pakietu lub został on usunięty. Należy pamiętać, iż w wersji testowej jest to możliwe.

Instalację pakietu można wykonać przez wyszukanie jego nazwy za pomocą opcji *Znajdź/Znajdź*. Należy wyszukać nazwę pakietu, następnie po podświetleniu pakietu nacisnąć znak +, po czym wykonać operacje literą *g*. W tym momencie pojawi się na ekranie lista pakietów potrzebnych do zainstalowania i niepotrzebnych, do usunięcia. Czynność zostanie wykonana po kolejnym wciśnięciu litery *g*. Należy jednak uważnie przeczytać listę pakietów, które zostaną usunięte przy instalacji. Możliwa jest sytuacja, w której instalacja przez zależności doprowadzi do usunięcia środowiska graficznego. Dlatego tez najlepiej korzystać konsekwentnie z jednego narzędzia zarządzającego pakietami. Pakiet można również wyszukać na liście pakietów. Do poruszania się po niej służą strzałki oraz klawisz *Enter*.

Jeżeli w trakcie instalacji pokażą się informacje o usuwaniu potrzebnych pakietów, nie należy przerywać procesu. Instalowanie pakietów jest czynnością czasochłonną, zależną od szybkości źródła, ale prostą. Naprawienie skutków przerwanego procesu usuwania pakietów może być nieco kłopotliwe. W przypadku usunięcia środowiska graficznego mało finezyjną, ale skuteczną metodą jest jego instalacja z użyciem narzędzia `tasksel`. Za jego pomocą można wybrać instalację środowiska graficznego obejmującą wszystkie potrzebne elementy.

Metodą ukierunkowaną na instalację jedynie niezbędnego oprogramowania jest instalacja dwóch pakietów: *kde-core* oraz *xserver-xorg-core*. Ich instalacja, niezależnie, czy za pomocą polecenia `aptitude` z parametrem, czy też przez okno instalacji, pozwoli na uruchomienie środowiska graficznego. Jednak środowisko takie będzie pozbawione wielu elementów, które należy zainstalować ręcznie.

Jeżeli dokładnie wiadomo, jak nazywa się pakiet do zainstalowania, można użyć polecenia `aptitude` z parametrem `install`, po czym podać nazwę pakietu. W analogiczny sposób można pakiety usuwać, zastępując `install` parametrem `remove`.

Często konieczne jest wyszukanie biblioteki wymaganej przez instalowany program niedostępny w formie pakietów lub też programu, którego pakiet zmienił nazwę. Pomocne w takiej sytuacji jest polecenie `apt-cache search nazwa`. Pozwala ono na przeszukiwanie nazw pakietów oraz ich opisów pod kątem podanego ciągu znaków.

Operacje na plikach

W systemie Linux właściwie wszystko jest plikiem. Można to sprawdzić, odczytując dowolny katalog za pomocą edytora *vim*. Okaże się, iż jest on plikiem z zapisanymi informacjami o położeniu w hierarchii. W linii poleceń wszystkie operacje plikowe ułatwia aplikacja MC (Midnight Commander). Można ją uruchomić przy użyciu polecenia `mc`. Pozwala ona na wykonywanie najpopularniejszych operacji za pomocą klawiszy funkcyjnych widocznych w tabeli 4.1. Nie jest jednak instalowana domyślnie.

Tabela 4.1. *Działanie przycisków funkcyjnych w mc*

Klawisz	Działanie
F1	Wyświetla pomoc programu.
F2	Wyświetla menu z dostępnymi czynnościami dla zaznaczonego pliku.
F3	Wyświetla podgląd wybranego pliku.
F4	Wczytuje plik w trybie edycji.
F5	Kopiuje plik.
F6	Przenosi plik.
F7	Tworzy katalog.
F8	Usuwa plik.
F9	Służy do przejścia do górnego menu.
F10	Kończy pracę programu.

Do poruszania się w programie służą przede wszystkim strzałki oraz klawisz *Tab*. Jedną z bardziej ułatwiających pracę funkcji jest możliwość rozpakowywania archiwów, dostępna w menu po naciśnięciu *F2*. Niezależnie, czy czynność ta jest wykonana za pomocą linii komend, czy też MC, system musi być wyposażony w aplikacje obsługujące dany format kompresji. Program ten może również nawiązywać połączenia FTP.

Pomimo dużych możliwości MC niektóre czynności wygodniej wykonywać za pomocą poleceń. Widoczne są one razem z odpowiadającymi im poleceniami w tabeli 4.2.

Tabela 4.2. *Polecenia operacji plikowych*

Polecenie	Działanie
`touch nazwa`	Tworzy plik o podanej nazwie.
`cp źródło cel`	Kopiuje plik ze źródła do wskazanego celu.
`rm nazwa`	Usuwa wskazany plik.
`rm -rf nazwa`	Usuwa wskazany katalog oraz jego zawartość.
`mv źródło cel`	Przenosi wskazany plik do określonego miejsca.
`ln -s źródło cel`	Tworzy dowiązanie symboliczne.

Kolejną istotną właściwością plików są prawa dostępu. Każdy plik posiada właściciela, grupę oraz zdefiniowane prawa dostępu. Wszystkie informacje o pliku, jakie są przechowywane, można zobaczyć za pomocą polecenia `ls -l`.

```
-rw------- 1 janko janko 44462 lis 2 17:58 prywatny
drw-r----- 5 janko janko  4096 lis 8 18:00 katalog_grupy
```

Na powyższym przykładzie widoczne są informacje dotyczące pliku o nazwie *prywatny* należącego do *janko*. Pierwsza sekcja po lewej stronie pokazuje informację, czy mamy do czynienia z plikiem, czy katalogiem, oraz jakie prawa dostępu obowiązują do pliku (katalog też jest plikiem). Litera na pierwszej pozycji oznacza katalog, kolejne trzy znaki oznaczają prawa właściciela, druga trójka — prawa grupy, a ostatnia trójka definiuje prawa pozostałych. Litera `r` oznacza odczyt, `w` — zapis, `x` — prawo do wykonania. W przypadku pierwszego pliku właściciel ma prawo odczytu i zapisu do pliku, w drugim przypadku takie samo prawo maja członkowie grupy. Po informacji o nazwie właściciela oraz grupy znajduje się objętość pliku. Można ją wyświetlić w bardziej przyjaznym formacie przez dodanie litery `h` do parametrów polecenia. Kolejną informacją jest data utworzenia pliku, na samym końcu zaś znajduje się nazwa pliku.

Przydatną umiejętnością jest przekazywanie wyniku jednego polecenia do kolejnego. W przypadku polecenia `ls` można dokonać takiego przekierowania w celu wyszukania pliku o konkretnej nazwie.

```
ls -l | grep nazwa_pliku
```

Powyższy przykład pokaże jedynie informacje o pliku, którego nazwa została wskazana. Filtrowania dokonuje program `grep`, do którego przekazane zostają dane z polecenia `ls`. Do samego przekazania danych między dowolnymi programami służy symbol `|`. Warunkiem przekazania jest, aby program przyjmujący dane spodziewał się formatu informacji przekazywanych przez program wcześniejszy.

Warto zwrócić uwagę, iż system Linux nie korzysta z rozszerzeń w formacie znanym z systemu Windows. Potrafi on ustalać rodzaj pliku bez jego zapisu w nazwie.

Często konieczne jest wyszukanie w systemie jakiegoś pliku. Wówczas z pomocą przychodzą dwa polecenia: `find` i `locate`. Najprostszą możliwością jest użycie polecenia `locate` z nazwą lub fragmentem nazwy pliku, który ma zostać odszukany. Bardziej złożonymi możliwościami dysponuje polecenie `find`. Pozwala ono na określenie katalogu, w którym rozpocznie się poszukiwanie, oraz bardziej elastyczne określenie nazwy poszukiwanego pliku.

```
find / -name '*plik???'
```

Powyższy przykład każe wyszukać plik składający się z dowolnej ilości znaków poprzedzających wyraz `plik` i zakończony trzema dowolnymi znakami po tym napisie. Możliwe jest podanie dowolnego katalogu jako początku wstukiwania. W powyższym przykładzie jest to katalog główny, czyli cały system.

Możliwe jest również wyszukiwanie plików według kryterium wielkości.

```
find /home/janko -size +10M
```

Powyższy przykład spowoduje wyszukanie plików większych niż 10 MB.

Kolejnym kryterium wyszukiwania plików może być nazwa użytkownika.

```
find / -user janko
```

Powyższy przykład pokaże wszystkie pliki w systemie należące do użytkownika *janko*.

Zarządzanie użytkownikami

System Linux został stworzony do wykorzystywania przez wielu użytkowników. Jego istotnym elementem jest dodawanie, usuwanie i przydzielanie uprawnień każdemu użytkownikowi systemu.

Do dodawania użytkowników służy polecenie `adduser`. Polecenie to należy uzupełnić nazwą użytkownika; utworzy ono konto wraz z katalogiem domowym oraz plikami konfiguracyjnymi. Zapyta również o hasło oraz kilka dodatkowych informacji, których podawanie nie jest konieczne.

Do skasowania użytkownika przeznaczone jest polecenie `usrdel -r nazwa`.

Z użytkownikami i ich grupami powiązane są prawa dostępu. Ważne jest, aby każdy użytkownik został przydzielony do grup umożliwiających mu dostęp do wszystkich potrzebnych plików i funkcji. Należy przy tym pamiętać, iż przydzielanie nadmiernych uprawnień nie tylko stanowi zagrożenie bezpieczeństwa rozumianego jako poufność i trwałość danych. Taka beztroska może także doprowadzić do sytuacji, w której zwykły użytkownik będzie mógł uszkodzić system operacyjny, w efekcie czego będzie on wymagał ponownej instalacji.

Lista istniejących grup jest zapisana w pliku */etc/group*. Dzięki grupom można ograniczyć dostęp użytkownika do funkcji posiadanych przez system, jak nagrywania płyt. Inną opcją jest umożliwienie wspólnego dostępu do plików. Nazwę użytkownika oraz grupy, do której należy dany plik, można odczytać dzięki poleceniu `ls -l`. Do ustawiania praw dostępu służy natomiast polecenie `chmod`.

Za pomocą polecenia `chmod` można zmieniać uprawnienia przez notację `chmod u+rwx nazwa_pliku`, gdzie `u` oznacza właściciela, wstawione w to miejsce `g` — grupę, zaś `o` — pozostałych. W analogiczny sposób można odebrać prawa przez zastąpienie znaku `+` znakiem `-`.

Przydatną możliwością jest dodawanie nowych grup, które pozwalają na zapewnienie dostępu do wybranego katalogu określonym użytkownikom. Do dodania grupy służy polecenie `groupadd nowa_grupa`, następnie w pliku */etc/group* można dopisać użytkowników do nowo powstałej grupy. Można również zmienić podstawową grupę użytkownika za pomocą polecenia `newgroup nazwa_użytkownika nazwa_grupy`. Do zmiany grupy, która jest właścicielem katalogu, służy polecenie `chgrp grupa katalog`.

Zarządzanie dyskami

Istotną umiejętnością jest zarządzanie dyskami. Na czynność tę składają się kontrola wolnego miejsca na dysku, montowanie dysków oraz nadawanie im uprawnień.

Do kontroli zainstalowanych nośników danych oraz miejsca na nich służy polecenie `df`. Samo polecenie korzysta z mało czytelnego formatu pojemności, bardziej przyjazny format można uzyskać przez zastosowanie parametru `-h` (listing 4.1).

Listing 4.1. *Przykładowy wynik polecenia df -h*

```
janko@Debian:~$ df -h
System plików          rozm    użyte   dost.  %uż   zamont. na
/dev/hda2              17G     4,2G    12G    27%   /
tmpfs                  253M    0       253M   0%    /lib/init/rw
udev                   10M     96K     10M    1%    /dev
tmpfs                  253M    0       253M   0%    /dev/shm
/dev/hda1              46M     27M     17M    61%   /boot
```

Powyższy przykład pokazuje, jak może wyglądać wynik polecenia `df`. Pokazane informacje zależą od wielkości dysku i wybranego podczas instalacji podziału na partycje. W pierwszej kolumnie znajduje się nazwa urządzenia; `hda` oznacza główny dysk, cyfry zaś oznaczają partycje. Kolejne dyski twarde otrzymują odpowiednie litery, np. `hdb`, `hdc`. Widoczne trzy urządzenia, `tmpfs`, `udev` i `tmpfs`, znajdują się wyłącznie w pamięci podręcznej i są wykorzystywane przez system.

Kolejne kolumny pokazują rozmiar urządzenia, ilość użytego miejsca, ilość wolnego miejsca oraz procentowe wykorzystanie przestrzeni. Ostatnia kolumna wskazuje miejsce w systemie, w którym dostępne jest urządzenie. Najważniejszy jest system plików zamontowany jako /. Znajduje się w nim cały system wraz z katalogami domowymi.

Innym przydatnym poleceniem pokazującym, jakie dyski twarde są obecne w systemie oraz na jakie partycje są podzielone, jest `fdisk -l`. Można dzięki niemu uzyskać dokładne dane dotyczące parametrów dysku, począwszy od ilości głowic, sektorów i cylindrów, a na typach partycji kończąc.

Ustawienia wszystkich zamontowanych dysków znajdują się w pliku */etc/fstab* (listing 4.2). Za jego pomocą można zmienić sposób montowania dysku, ograniczyć prawa dostępu do niego lub dodać nowy dysk. Nie należy w pliku tym wprowadzać zmian odnoszących się do urządzeń istniejących po instalacji systemu bez dokładnej wiedzy o skutkach. Umiejętność dodawania nowych wpisów do pliku będzie przydatna w przypadku dołożenia do systemu dodatkowego dysku twardego.

Listing 4.2. *Plik /etc/fstab*

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda2       /               ext3    errors=remount-ro 0       1
/dev/hda1       /boot           ext3    defaults        0       2
/dev/hda5       /home           ext3    defaults        0       2
/dev/hda6       none            swap    sw              0       0
/dev/hdb1       /dokumenty      ext3    auto,user,rw        0       2
/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
```

Pierwsze dwie kolumny mówią o tym, jakie urządzenia i gdzie mają być zamontowane. Nazwa urządzenia zależy od tego, gdzie jest podłączone; w przypadku partycji decyduje jej pozycja. Punkty montowania natomiast pozwalają na odczytanie danych znajdujących się na urządzeniach. Trzecia kolumna zawiera informacje o rodzaju systemu plików. W przypadku nowego dysku do jego sprawdzenia można użyć polecenia `fdisk -l`. Odczyta ono informacje z niezamontowanego jeszcze dysku. Kolumna piąta zawiera informacje potrzebne przy tworzeniu kopii bezpieczeństwa, zaś szóstą wykorzystuje polecenie `fdisk -l`. Na jej podstawie program ten ustala, czy i w jakiej kolejności sprawdzić poprawność partycji. Pominięta kolumna czwarta zawiera opcje montowania pozwalające kontrolować sposób montowania czy dostęp do dysku. Opis możliwości znajduje się w tabeli 4.3.

Tabela 4.3. *Opcje montowania pliku fstab*

Opcje	Znaczenie
`auto`; `noauto`	Opcja `auto` sprawi, iż urządzenie będzie montowane automatycznie podczas startu systemu. Jeżeli ma ono być montowane przez użytkownika, należy wybrać `noauto`.
`user`, `nouser`	Opcja `user` pozwala montować urządzenie zwykłemu użytkownikowi. Użycie opcji `nouser` da taką możliwość tylko administratorowi.
`exec`, `noexec`	Opcja `exec` zezwala na wykonywanie plików binarnych znajdujących się na partycji.
`ro`, `rw`	Opcja `ro` sprawia, że pliki na urządzeniu są tylko do odczytu, `rw` pozwala na ich odczyt i zapis.
`sync`, `async`	Opcja ta pozwala ustawić tryb pracy nośnika danych. W przypadku `sync` jest to tryb synchroniczny, zapis fizyczny na nośniku jest dokonywany razem z wydaniem polecenia zapisu. Opcja `async` powoduje, iż zapis może być wykonany po wydaniu polecenia. W efekcie jeżeli urządzenie zostanie w niewłaściwy sposób odłączone, dane zostaną utracone.
`defaults`	Opcja ta jest sposobem jednoczesnego ustawienia kilku opcji.

Dyski można również montować ręcznie za pomocą polecenia `mount`. Należy w tym celu podać rodzaj systemu plików, urządzenie oraz katalog docelowy. Jako przykład można zamontować dysk ustawiony jako drugi na pierwszym kanale IDE.

```
mount -t ext3 /dev/hdb1 /mnt/dysk
```

Jest to jedna partycja z systemem plików *ext3* dostępna po zamontowaniu w katalogu */mnt/dysk*. Aby zdemontować urządzenie, należy wykorzystać polecenie `umount` z parametrem w postaci katalogu docelowego.

Linux pozwala również na kontrolowanie parametrów pracy dysku twardego. Aby uzyskać informacje i wpływ na pracę urządzenia, należy zainstalować program `hdparm`. Do jego wykorzystania potrzebne są uprawnienia konta root, w razie ich nieposiadania użytkownik otrzyma informację o braku takiego polecenia.

```
hdparm /dev/hda
```

Wywołanie polecenia `hdparm` wymaga wskazania dysku, o którym mają zostać udzielone informacje. Najciekawszym zastosowaniem tego narzędzia nie jest jednak wyświetlanie parametrów, lecz ich modyfikacja. Pełen opis możliwości jest dostępny w opisie polecenia po wpisaniu komendy `man hdparm`. Można na przykład zmniejszyć głośność pracy dysku przez zmniejszenie ilości obrotów.

```
hdparm -M /dev/hdb
/dev/hdb:
acoustic               =128 (128=quiet ... 254=fast)
```

Na powyższym przykładzie widać, jak wywołać informacje o obecnym trybie pracy dysku i trybach dostępnych. W tym przypadku jest to tryb cichy, tym samym wolny. Starsze dyski twarde posiadają tylko dwa tryby: wolny i szybki.

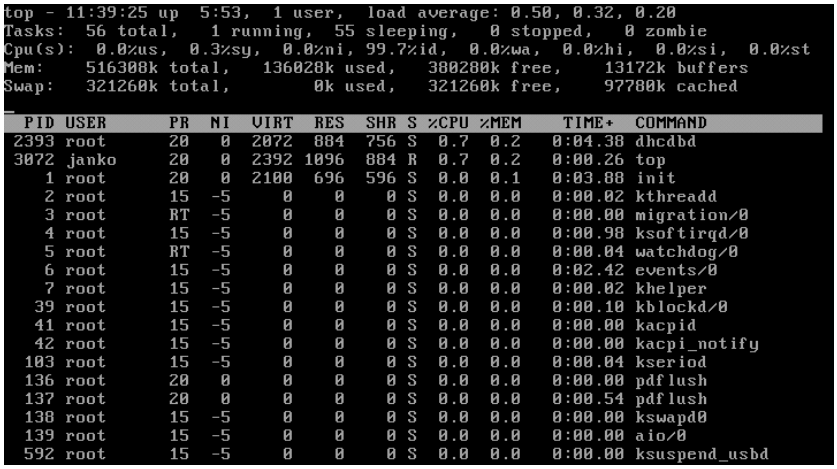
Wpływ takich zmian na szybkość pracy można sprawdzić, korzystając z parametru `-Tt`. Pozwala on na wykonanie pomiaru szybkości przesyłu danych; należy pamiętać, aby wykonać taki pomiar kilkukrotnie przed wyciąganiem wniosków.

Zarządzanie procesami

Każda uruchomiona przez użytkownika aplikacja funkcjonuje w systemie jako proces, który wykorzystuje dostępne zasoby. Programy z wykorzystaniem konsoli mogą być uruchamiane w sposób blokujący konsolę lub w tle, co pozwala na dalszą pracę w konsoli. W drugim przypadku procesy mogą pracować nawet po wylogowaniu się użytkownika. Zdarzają się również sytuacje, w których proces staje się tak zwanym „zombi” — procesem, który zużywa całą dostępną moc obliczeniową bez możliwości automatycznego wyłączenia go przez system. Wyjściem z takich sytuacji jest wyłączenie systemu lub wykorzystanie uprawnień administratora. Nieograniczone uprawnienia pozwalają zarówno przeglądać procesy wszystkich użytkowników, jak i przerywać ich działanie. Jednak konieczna jest do tego znajomość narzędzi monitorujących pracę procesów oraz służących eliminowaniu tych, które są zbędne.

Widoczna jest tu przewaga konsoli nad środowiskiem graficznym. W przypadku uruchomienia przez użytkownika wielu procesów zachowujących się jak zombi, ograniczona ilość zasobów spowoduje problemy w pracy z środowiskiem graficznym. Konsola jako mniej wymagająca pozostanie sprawniejszym narzędziem, które nie będzie w tak dużym stopniu odczuwać ograniczenia dostępnych zasobów.

Podstawowym poleceniem służącym kontroli procesów jest `top`. Pozwala ono obserwować w trybie ciągłym pracujące procesy i stopień wykorzystania zasobów systemu. Jak widać na rysunku 4.2, program ten podaje bardzo dużą ilość informacji.



```
top - 11:39:25 up  5:53,  1 user,  load average: 0.50, 0.32, 0.20
Tasks:  56 total,   1 running,  55 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    516308k total,   136028k used,   380280k free,    13172k buffers
Swap:   321260k total,        0k used,   321260k free,    97780k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2393 root      20   0  2072  884  756 S  0.7  0.2   0:04.38 dhcdbd
 3072 janko     20   0  2392 1096  884 R  0.7  0.2   0:00.26 top
    1 root      20   0  2100  696  596 S  0.0  0.1   0:03.88 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:00.98 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.04 watchdog/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:02.42 events/0
    7 root      15  -5     0    0    0 S  0.0  0.0   0:00.02 khelper
   39 root      15  -5     0    0    0 S  0.0  0.0   0:00.10 kblockd/0
   41 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   42 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  103 root      15  -5     0    0    0 S  0.0  0.0   0:00.04 kseriod
  136 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  137 root      20   0     0    0    0 S  0.0  0.0   0:00.54 pdflush
  138 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kswapd0
  139 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  592 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ksuspend_usbd
```

Rysunek 4.2. *Widok procesów przez polecenie top*

Poczynając od lewego górnego rogu okna, `top` wyświetla obecny czas, czas pracy systemu, ilość zalogowanych użytkowników oraz średnie obciążenie. Informacje te można również otrzymać za pomocą polecenia `uptime`. Warto przyjrzeć się średniemu obciążeniu. Składa się ono z trzech wartości odpowiadających 1 minucie, 5 minutom i 15 minutom. Nie przedstawiają one procentowej wartości wykorzystanego czasu procesora, lecz ilość procesów, które chciały z niego korzystać. W obliczeniu tej wartości brane są pod uwagę wykonywane procesy i procesy czekające na wykonanie.

Druga linia zawiera informacje o ilości procesów, procesy działające, uśpione, zatrzymane i zombi. Kolejna linia wskazuje, jak wykorzystany jest czas procesora. Objaśnienie skrótów widoczne jest w tabeli 4.4.

Tabela 4.4. *Opis kategorii procesów*

Kategoria procesów	Opis
us	czas CPU przeznaczony na wykonanie procesów użytkownika
sy	czas CPU przeznaczony na wykonanie procesów systemu
ni	czas CPU przeznaczony na wykonanie procesów o zmienionym priorytecie
wa	ilość czasu CPU poświęcona na operacje wejścia-wyjścia (ang. I/O)
hi	czas CPU poświęcony przerwaniom sprzętowym
si	czas CPU poświęcony przerwaniom programowym
st	ilość czasu „ukradzionego” przez maszynę wirtualną

Linia czwarta odnosi się do wykorzystania pamięci RAM. Widać kolejno, jaka ilość pamięci jest: zainstalowana w systemie, wykorzystana, wolna oraz przeznaczona na bufor. Kolejna linia odpowiada za pamięć wymiany znajdującą się na dysku twardym. Kolejność i znaczenie informacji na jej temat jest takie samo jak w przypadku pamięci RAM.

Dalsze linie poświęcone są pracującym procesom. Kolumny oznaczone są skrótami symbolizującymi parametry procesu. Zostały one objaśnione w tabeli 4.5.

Tabela 4.5. *Opis podglądu procesów*

Skrót parametru	Znaczenie parametru
PID	identyfikator procesu
USER	użytkownik procesu
PR	priorytet procesu
NI	modyfikator priorytetu procesu
VIRT	całkowita pamięć podręczna wykorzystywana przez proces; kod + dane + biblioteki + pamięć wymiany
RES	ilość pamięci RAM zajętej przez proces
SHR	pamięć współdzielona między kilkoma procesami
S	status procesu
%CPU	procentowa ilość czasu procesora wykorzystana przez proces
%MEM	procentowa ilość pamięci RAM wykorzystana przez proces
TIME+	ilość czasu procesora wykorzystanego przez proces od jego uruchomienia
COMMAND	nazwa programu

Identyfikator procesu pozawala na zakończenie nieporządnego procesu za pomocą polecenia `kill`. Polecenie to przyjmuje jako parametr identyfikator procesu oraz rodzaj sygnału, jaki ma wysłać. Tą metodą można zarządzić zamykanie procesu lub przerwać proces niezależnie od jego stanu.

```
kill -s 9 4858
```

Powyższy przykład polecenia `kill` spowoduje natychmiastowe zakończenie procesu o numerze identyfikacyjnym `4858`.

Warto zwrócić uwagę na powiązanie między priorytetem procesu a modyfikatorem priorytetu. Każdy proces uruchamiany jest z takim samym priorytetem, zapewniającym równy dostęp do czasu procesora. Za pomocą `NI` możliwa jest zmiana priorytetu. Dobrym przykładem na działanie tego mechanizmu jest program BOINC, przeznaczający moc obliczeniową procesora na rzecz badań naukowych. Po zainstalowaniu procesy przez niego uruchamiane będą otrzymywały `NI=19`. Dzięki temu pomimo że zużyją całą dostępną moc obliczeniową, zawsze będą ustępowały miejsca procesom o wyższym priorytecie. Ustawienie za pomocą `NI` priorytetu na wyższy spowodowałoby brak reakcji systemu na jakiekolwiek działania użytkownika. Wyjaśnienia wymaga kwestia wielkości priorytetu. Wyższy priorytet rozumiany jako pierwszeństwo w dostępie do czasu procesora oznaczony jest w systemie Linux jako mniejsza liczba. Proces z `PR=15` będzie miał pierwszeństwo przed procesem z `PR=20`.

Priorytet procesu można zmienić dzięki poleceniu `renice`. W tym celu konieczna jest znajomość identyfikatora procesu, można również zmienić priorytet wszystkich procesów należących do danego użytkownika. Jednak uzyskanie potrzebnych danych przy użyciu wyłącznie polecenia `top` może być problematyczne. Statyczną listę wszystkich procesów wraz z informacją o wykorzystywanych przez nie zasobach można uzyskać za pomocą polecenia `ps` z parametrem `aux` lub `-elf`. Każdy z podanych parametrów wyświetla nieco inne informacje o procesach. Do wyświetlenia procesów jednego użytkownika wystarczy zmienić parametr na `-U nazwa_uzytkownika`.

```
ps -e
...
2585 ?          00:00:16 kadu
...
kill 2585
```

Powyższe przykładowe polecenia wyświetlą listę wszystkich procesów w systemie, a następnie zamkną komunikator Kadu.

Znając identyfikator Kadu, zamiast zamykać program, można zmienić jego priorytet.

```
renice +1 2585
```

Powyższe polecenie spowoduje nieznaczne obniżenie priorytetu procesu. Z powyższych przykładów nie należy wyciągać wniosku, iż procesy zawsze mają ten sam identyfikator. Uruchamiany proces za każdym razem otrzymuje nowy identyfikator.

Po nabyciu umiejętności kończenia pracy procesów metodą awaryjną, jaką jest polecenie `kill`, można zacząć korzystać z procesów działających w tle. Pracując w konsoli, można znaleźć programy niemal do każdego celu, prócz oglądania filmów. Jednak uruchomienie każdego programu powoduje zablokowanie konsoli na jego użytek. Wyjściem z takiej sytuacji jest uruchomienie programów niewymagających interakcji w tle. Przykładem jest *mpg321* służący do odtwarzania muzyki. Program ten uruchomiony normalnie pozwoli na słuchanie muzyki, jednak niemożliwe będzie wykorzystanie konsoli w innym celu. Rozwiązaniem jest uruchomienie odtwarzacza w tle, służy temu znak ampersand.

```
mpg321 -z muzyka.mp3 &
```

Do wyświetlenia procesów pracujących w tle służy polecenie `jobs -l`. Pokazuje ono również identyfikator procesu, dzięki czemu możliwe jest jego zakończenie lub przesunięcie na pierwszy plan. Do przesunięcia na pierwszy plan służy polecenie `fg`, wymaga ono podania numeru procesu lub jego nazwy. Nie należy mylić numeru procesu wyświetlanego przez polecenie `jobs` z identyfikatorem procesu.

Proces będący na pierwszym planie można zakończyć kombinacją klawiszy *Ctrl+C* lub zatrzymać kombinacją *Ctrl+Z*. Zatrzymany proces można wznowić za pomocą polecenia `fg`. Niezależnie, czy odnosi się ono do procesu wstrzymanego, czy pracującego w tle, powinno przyjąć formę widoczną poniżej.

```
fg %numer_procesu
```

Proces zatrzymany w analogiczny sposób można uruchomić w tle, zastępując polecenie `fg` przez `bg`.

Cron

Program *cron* pozwala na uruchamianie programów w wyznaczonym czasie. Programy takie można pisać samemu, w najprostszym przypadku są to pojedyncze polecenia powłoki. Inną możliwością jest tworzenie skryptów powłoki, czyli plików zawierających zbiory poleceń. Skrypty mogą być wykonywane jako czynności administratora lub

użytkownika. Dlatego też każdy użytkownik dysponuje własną listą zadań. Do edycji listy zadań wykonywanych przez *cron* służy polecenie `cron -e`. Powoduje ono otworzenie pliku, w którym należy podać czas wykonania zadania oraz polecenie.

```
0 * * * * /bin/ls
```

Powyższy przykład pokazuje wygląd wpisu do pliku, który spowoduje uruchamianie polecenia `ls` równo o każdej godzinie. Pierwsze pole z cyfrą zero oznacza liczbę minut po pełnej godzinie, kolejne gwiazdki oznaczają: godzinę, dzień miesiąca, miesiąc i dzień tygodnia. Gwiazdka oznacza, iż każda wartość jest prawidłowa. Możliwe jest również zdefiniowanie kilku wybranych godzin czy dni, w których program ma być uruchamiany:

```
* 12,14,16 * * * /bin/ls
```

Widoczny powyżej przykład spowoduje uruchamianie wskazanego polecenia o godzinie 12, 14 i 16. W podobny sposób można zdefiniować przedział czasowy uruchamiania programu poprzez połączenie dwóch wartości myślnikiem.

Przeglądanie już ustawionych zadań jest możliwe dzięki poleceniu `crontab -l`. Spowoduje ono wyświetlenie wszystkich zadań użytkownika. Administrator może przeglądać zadania wszystkich użytkowników dzięki poleceniu z parametrem `-u`.

```
crontab -u nazwa_uzytkownika -l
```

Tworzenie kopii zapasowych

Tworzenie kopii zapasowych jest istotne niemal dla każdego. Argumentów nie trzeba szukać daleko. W trakcie pisania tej książki jedynie dzięki szczęściu awaria dysku twardego nie spowodowała zniszczenia jedynej kopii kilku rozdziałów. Szczęście to było wynikiem utworzenia dzień wcześniej kopii na inny komputer w celu ułatwienia pracy.

Nie sposób przewidzieć czasu i przyczyny awarii, która może doprowadzić do utraty danych. Linux posiada kilka narzędzi mających na celu ułatwienie tworzenia kopii zapasowych. Jednak znacznie większą wartość ma utworzenie własnego, chociażby najprostszego skryptu tworzącego kopie zapasowe. Ich podstawą będzie albo skrypt uruchamiany przy włączaniu i wyłączaniu systemu, albo omówiony *cron*.

Podstawą do uruchomienia skryptów w czasie startu i zamykania systemu jest katalog */etc/init.d*. W nim znajdują się skrypty uruchamiające wiele usług. Nie należy jednak plików tam obecnych usuwać dla pozbycia się usługi. Czynność taka spowoduje kłopoty przy próbie ich ponownego uruchomienia. Poradzić sobie z tym można w analogiczny sposób do dodawania i usuwania własnych skryptów. Najpierw jednak skrypt taki trzeba napisać.

Na listingu 4.3 widać prosty, ale jednocześnie uczący bardzo wielu rzeczy skrypt. Przed jego uruchomieniem konieczne jest omówienie wszystkich linii krok po kroku.

W linii pierwszej znajduje się informacja, z której powłoki systemu korzysta skrypt. W przypadku skryptów tego rodzaju jest to powłoka *sh* (ang. *Shell*).W normalnej pracy użytkownik Debiana korzysta z powłoki *bash* (ang. *Burn Again Shell*).

Listing 4.3. *Kopie bezpieczeństwa przy starcie i zamykaniu systemu*

```
1: #! /bin/sh
2: #/etc/init.d/backup
3: case "$1" in
4:       start)
5:       echo "Wykonuje kopie przy starcie systemu"
6:       mount -t smbfs -o username=nobody, password= 192.168.1.2:dane /mnt/samba
7:       cp -rf /mnt/samba/dane/* /home/janko/dane &
8:       ;;
9:       stop)
10:      echo "Wykonuje kopie przy zamykaniu systemu
11:      cp -rf /home/janko/dane/* /mnt/samba/dane
12:      ;;
13:      *)
14:      echo "Usage: /etc/init.d/ramdisk {start|stop}"
15:      exit 1
16:      ;;
17: esac
18: exit 0
```

Linia jest elementem języka skryptowego wykonywanego przez powłokę; `case` oznacza przypadek, a `$1` odnosi się do pierwszego ciągu znaków, jaki zostanie podany po wywołaniu skryptu. Jeżeli tym ciągiem znaków będzie wyraz `start`, wykonane zostaną polecenia znajdujące się w liniach od 5. do 7. włącznie.

Linia 5. spowoduje wyświetlenie na monitorze tekstu znajdującego się w cudzysłowie. Linia 6. może sprawić na początku nieco problemów. Powoduje ona zamontowanie udostępnionego za pomocą programu *Samba* dysku w lokalnym katalogu */mnt/samba*. Kolejne elementy tego polecenia oznaczają system plików *smbfs*, nazwę użytkownika i hasło oraz położenie udostępnionego dysku w sieci. W przypadku braku możliwości użycia drugiego komputera można wybrać dowolne miejsce w lokalnym systemie. Dobrze jednak, gdy dane znajdują się na dwóch różnych dyskach.

Linia 7. jest niczym więcej jak kopiowaniem plików. Jednym elementem, który może zwrócić uwagę, jest znak ampersand na końcu linii. Dzięki niemu uruchamianie systemu nie zostaje zatrzymane podczas kopiowania plików. Linia 8. kończy działania wykonywane w pierwszym przypadku.

W linii 9. rozpoczyna się druga część skryptu wykonywana przy zamykaniu systemu. Jest ona dużo prostsza, powoduje skopiowanie danych z powrotem do miejsca, z którego zostały pobrane.

W liniach od 13. do 15. znajduje się jeszcze jedna sekcja. Jest ona wywoływana w przypadku podania skryptowi jakiegokolwiek innego parametru niż zdefiniowane wcześniej.

Tak stworzonemu plikowi umieszczonemu w katalogu */etc/init.d* należy nadać prawa do wykonania oraz dołączyć do sekwencji uruchamiania systemu.

```
chmod +x backup
update-rc.d backup default
```

Po wykonaniu powyższych poleceń przy każdym starcie i wyłączeniu systemu wykonywany będzie stworzony wcześniej skrypt. Aby go usunąć, należy zmodyfikować nieznacznie polecenie, zastępując `default` przez `remove`.

Drugą możliwością tworzenia kopii zapasowych jest wykorzystanie omówionego programu *cron*. Wystarczy w tym celu nieznacznie zmodyfikować napisany wcześniej skrypt.

Skrypt na listingu 4.4 będzie wykonywał kopię wskazanych danych, a następnie łączył je w plik *tar*, który zostanie skompresowany za pomocą programu *bzip2*. Działanie skryptu zakończy usunięcie niespakowanej kopii plików.

Listing 4.4. *Skrypt kopiujący dla programu cron*

```
1: #! /bin/sh
2: #/etc/init.d/backup
3:       cp -rf /home/janko/dane/* /media/dysk/dane
4:       tar -cvf /media/dysk/spakowane/polaczone_dane.tar /media/dysk/dane/*
5:       bzip2 /media/dysk/spakowane/polaczone_dane.tar
6:       rm -rf /media/dysk/dane/*
```

Skrypt ten jest uproszczony do granic możliwości, wykona on jednak swoje zadanie. Należy go dodać do zadań programu *cron* z odpowiednią częstotliwością wykonania.

Konfiguracja systemu

System Linux pozwala na konfigurację każdego elementu swojej pracy, poczynając od głębokości kolorów wyświetlanego obrazu, a na szybkości pracy dysku kończąc. Zależnie od konfigurowanego aspektu pracy systemu, w większości przypadków wystarcza zmiana wpisu w pliku konfiguracyjnym. Niektóre czynności można jednak wykonać przez dodanie skryptów uruchamiających się podczas startu systemu.

Konfigurację można zacząć od zmiany tekstu powitalnego po załogowaniu się użytkownika w konsoli. Wiadomość tę można znaleźć i zmienić w pliku */etc/motd*. W pliku tym znajduje się tekst wyświetlany po zalogowaniu użytkownika.

Właściwe pliki konfiguracyjne odnoszące się do funkcjonowania powłoki *bash*, czyli naszego trybu tekstowego, są podzielone na dwie kategorie: pliki z ustawieniami o zasięgu globalnym oraz lokalnym. Jeżeli jakiś aspekt pracy środowiska jest zdefiniowany w obu rodzajach plików, znaczenie ma konfiguracja lokalna, czyli użytkownika.

Do umiejętności wprowadzania zmian w konfiguracji konieczna jest znajomość podstaw języka skryptowego, jakim jest *bash*. Pierwszym elementem są zmienne, stanowią one pojemniki do przechowywania danych. Definicja zmiennej wymaga podania jej nazwy, znaku równości oraz wartości, która będzie przechowywana. Podobnie jak istnieją pliki o zasięgu lokalnym i globalnym, tak również zmienne dzielą się na lokalne i globalne. Należy również pamiętać, iż dwie takie same nazwy, ale napisane przy użyciu dużych i małych liter, oznaczają inne zmienne. Zmienną można wyświetlić za pomocą polecenia `echo` oraz nazwy zmiennej:

```
echo $SHELL
```

Drugą ważną rzeczą są komentarze w plikach konfiguracyjnych. Każda linia poprzedzona znakiem # nie jest wczytywana przez system. Prócz opisów wewnątrz pliku pozwala to na czasowe eliminowanie ustawień bez ich trwałego usuwania z pliku.

Spróbujmy przeanalizować pliki konfiguracyjne. Pierwszym z nich o zasięgu globalnym jest */etc/profile* (listing 4.5).

Listing 4.5. *Plik /etc/profile*

```
1:# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
2:# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
3:if [ "`id -u`" -eq 0 ]; then
4:  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
5:else
6:  PATH="/usr/local/bin:/usr/bin:/bin:/usr/games"
7:fi
8:if [ "$PS1" ]; then
9:  if [ "$BASH" ]; then
10:    PS1='\u@\h:\w\$ '
11:  else
12:    if [ "`id -u`" -eq 0 ]; then
13:      PS1='# '
14:    else
15:      PS1='$ '
16:    fi
17:  fi
18:fi
19:export PATH
20:umask 022
```

Pierwsze dwie linijki pliku zawierają jedynie informacje dla użytkownika, o jego funkcji. W Linii 3. następuje sprawdzenie identyfikatora użytkownika; jeżeli jest równy `0`, wówczas ustawiana jest widoczna w linii 4. zawartość zmiennej `PATH`. Obecny identyfikator można zobaczyć, wydając w konsoli polecenie `id`. Zmienna `PATH` określa położenie programów, których wywołanie będzie wymagało podania jedynie nazwy, bez pełnej ścieżki do miejsca, w którym się znajduje. Jeżeli warunek nie zostanie spełniony, zmienna `PATH` dla logującego się użytkownika zostanie ustawiona w linii 6. Znaczenie tego warunku staje się zrozumiałe po wyświetleniu w konsoli identyfikatora konta root. Jedynie to konto spełnia pierwszy warunek, otrzymując dostęp do wszystkich poleceń. Korzystając z tego, można odebrać zwykłym użytkownikom posiadany dostęp lub go zwiększyć.

Kolejny warunek, w linii 8., sprawdza, czy istnieje zmienna `PS1`. Zawiera ona informacje, co ma się znaleźć w konsoli po lewej stronie kursora. Kolejna linia zawiera sprawdzenie istnienia zmiennej `BASH`; jeżeli istnieje, w linii 10. ustawiona zostaje zawartość zmiennej `PS1`. Każda litera w podanym ciągu ma określone znaczenie, przedstawione w tabeli 4.6.

W dalszych liniach zdefiniowane są inne wartości zmiennej `PS1`, w razie niespełnienia warunków. Istotna jest linia 19.; samo ustawienie zmiennej nie wystarcza, aby system mógł ją wykorzystać. Zmienną taką należy wyeksportować.