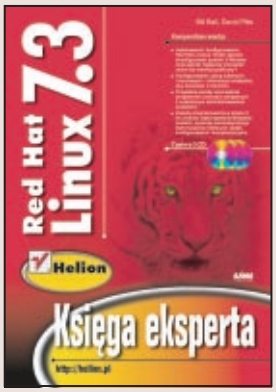
Red Hat Linux 7.3. Księga eksperta



Autor: Bill Ball
Tłumaczenie: Maciej Pasternacki
ISBN: 83-7197-787-5
Tytuł oryginału: Red Hat Linux Unleashed
Format: B5, stron: 746

Red Hat to jedna z najwcześniej powstałych i najpopularniejszych dystrybucji Linuksa. Książka „Red Hat Linux 7.3. Księga eksperta" to kompletne omówienie tej dystrybucji. Znajdziesz w niej wszystkie informacje niezbędne, aby zainstalować i skonfigurować Red Hat Linux, jak również nim administrować oraz przebudować go w optymalny sposób. Najpierw dowiesz się, jak przygotować i zaplanować instalację.
Po wskazówkach na temat konfiguracji nastąpi wprowadzenie do administracji systemu, a następnie opis zaawansowanych technik administracyjnych. W zdobywaniu nowych umiejętności pomoże Ci także rozdział o oprogramowaniu użytkowym i programowaniu.

W książce zostały omówione:

- Nowości w Red Hat Linux 7.3
- Przygotowanie do instalacji, instalację i konfigurację systemu
- Praca w systemie X Window
- Zarządzanie usługami i instalacja oprogramowania
- Zarządzanie użytkownikami
- Praca z dyskami i systemami plików w Linuksie
- Konfigurowanie Linuksa do pracy w sieci i w Internecie
- Serwer Apache i serwery baz danych MySQL i PostgreSQL, serwery FTP
- Konfigurowanie usług poczty elektronicznej i serwera grup dyskusyjnych
- Programowanie: C/C++, skrypty powłoki i Perl
- Programy użytkowe: StarOffice, KDE Office i GNOME

Księga eksperta, wbrew nazwie, nie jest przeznaczona wyłącznie dla ekspertów, znających na wylot Red Hat-a, choć i oni mogą z niej skorzystać w celu zapoznania się ze zmianami wprowadzonymi w wersji tej dystrybucji. Jest to także pozycja dla tych, którzy dopiero chcą się stać takimi ekspertami. To książka, w której znajdziesz wszystko, co potrzebne do pracy z systemem Red Hat Linux 7.3.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 10.
Zarządzanie systemami plików

W tym rozdziale:

- historia systemów plików Linuksa,
- działanie dysku,
- tablica partycji,
- pliki,
- nazewnictwo urządzeń blokowych i znakowych,
- tworzenie urządzeń przy użyciu polecenia mknod,
- urządzenia znakowe, blokowe i specjalne,
- systemy plików aktualnie dostępne w systemie,
- systemy plików obsługiwane przez Linuksa,
- alternatywne systemy plików Linuksa,
- systemy plików DOS,
- systemy plików CD-ROM,
- tworzenie systemów plików,
- montowanie systemów plików,
- automatyczne montowanie — plik fstab,
- macierze RAID,
- przenoszenie systemu plików,
- LVM,
- narzędzia graficzne do montowania systemów plików,
- przykłady,
- optymalizacja działania dysku,
- optymalizacja działania systemów plików.

Jednym z najczęściej źle rozumianych określeń technicznych związanych z Linuksem jest system plików. Zrozumienie, czym właściwie jest system plików może przynieść prawdziwe oświecenie.

System plików jest czymś więcej niż formatem dysku twardego lub dyskietki, czymś więcej niż sam dysk lub proces formatowania; jest on zdefiniowany jako struktura i zorganizowanie danych na urządzeniu je przechowującym. Innymi słowy, jest to sposób, w jaki pliki są przechowywane i ułożone na dysku (albo taśmie, płycie CD-ROM lub innym medium). Uniksowe systemy plików, które nas najbardziej interesują, służą do przechowywania nie tylko danych, ale też metadanych (właściwości danych — właściciel, prawa dostępu i inne dodatkowe informacje).

Projektanci systemów plików zawsze wydają się wiedzieć lepiej, co składa się na dobry system plików. Niektóre z systemów plików są tworzone, aby posunąć obecną technologię o krok dalej; inne powstają jako systemy zastrzeżone w celu osiągnięcia przewagi nad konkurencją lub wprowadzenia funkcjonalności obecnej jedynie w danym systemie. Zwykle użytkownicy jakiegoś systemu operacyjnego muszą korzystać z systemu plików właściwego dla niego; nawet komercyjne systemy Unix używają zastrzeżonych systemów plików i zastrzeżonych schematów partycjonowania dysku. Jest to nieprzyjazne i powoduje niezgodności, ale z biznesowego punktu widzenia ma swój sens. Innym, być może bardziej znanym przykładem, jest fakt, że użytkownicy systemu Windows 98 mogą używać tylko systemów plików FAT Microsoftu. W systemie Windows można zainstalować sterowniki z obcego źródła i używać innych systemów plików, ale sam system musi być uruchamiany z systemu plików Microsoftu. Użytkownicy Linuksa mają większą swobodę, mogą wybierać spośród kilku systemów plików — niektóre z nich zapewniają kompatybilność wstecz, inne współpracę z innymi systemami operacyjnymi, a jeszcze inne udostępniają najnowocześniejszą funkcjonalność, obecną w najlepszych komercyjnych systemach operacyjnych. Możliwość korzystania z różnych systemów plików jest silną stroną Linuksa.

System plików Linuksa jest bardzo skomplikowany, ale wszystkie jego funkcje albo są wzorowane na funkcjonalności dostępnej w komercyjnych systemach uniksowych, albo po prostu uważane przez twórców Linuksa za przydatne. Na szczęście własny system plików Linuksa jest odporny na błędy i działa bardzo dobrze z domyślnymi ustawieniami. Red Hat zapewnił wygodne, łatwe w użyciu narzędzie konfiguracyjne używane podczas instalacji, ale późniejsza konfiguracja systemów plików jest przeważnie wykonywana ręcznie (albo wcale). Ciekawy czytelnik na pewno znajdzie mnóstwo opcji do zabawy.

Historia systemów plików Linuksa

Linux był tworzony na podstawie wzorowanego na Uniksach systemu operacyjnego minix (innymi słowy, Linus Torvalds użył miniksa do kompilacji Linuksa), więc pierwotnie Linux korzystał dla wygody z systemu plików miniksa. Przestało to być wygodne, gdy potrzeby Linuksa przerosły ograniczenia i możliwości pożyczonego systemu plików. W systemie plików miniksa największy możliwy rozmiar pliku był równy 64 MB, wpisy w katalogach miały stały rozmiar, a maksymalna długość nazwy pliku wynosiła 14 znaków.

Z tymi ograniczeniami uporano się, dodając do jądra systemu warstwę wirtualnego systemu plików (VFS — *Virtual File System*). VFS umożliwiał obsługę różnych systemów plików ze sterownikami ładowanymi jako moduły jądra. Możliwość użycia sterowników różnych systemów plików sprawiała, że dowolny system plików na dysku dla systemu operacyjnego wygląda tak samo. Po załadowaniu odpowiedniego modułu system jest w stanie odczytywać i zapisywać pliki zapisane na dysk przy użyciu prawie każdego systemu plików.

Oto spis modułów jądra obsługujących poszczególne systemy plików w Red Hat Linuksie 7.3.:

- adfs — Acorn Disk filing system*,
- affs — systemy plików komputerów Amiga*,
- bfs — *Boot Filesystem* używany w systemie SCO Unixware,
- coda — sieciowy system plików Carnegie Mellon University,
- cramfs — skompresowany system plików dla kości ROM,
- efs — system plików starych wersji systemu SGI IRIX, używany także w starych płytach CD-ROM*,
- ext2 — rozszerzony system plików Linuksa wersja 2 (nadal standardowy),
- ext3 — rozszerzony system plików Linuksa wersja 3 (dodanie żurnalu do systemu ext2),
- fat — ogólna obsługa systemów plików Microsoftu z rodziny FAT,
- freevxfs — darmowa wersja systemu plików Veritas VxFS używany w SCO Unixware, Sun Solaris i HP-UX,
- hfs — system plików systemu MacOS,
- hpfs — system plików systemu OS/2*,
- intermezzo — sieciowy system plików z możliwością pracy *offline*,
- iso9660 — system plików dla płyt CD-ROM,
- jfs — IBM Journaled Filesystem,
- minix — system plików miniksa,
- msdos — odmiana systemu plików FAT używana w systemie MS-DOS (FAT12, FAT16),
- ncpfs — sieciowy system plików Novell Netware,
- nfs — uniksowy sieciowy system plików,
- ntfs — system plików Windows NT (tylko do odczytu)*,
- qnx4 — system plików systemów operacyjnych QNX4 i QNX6*,
- reiserfs — system plików z żurnalem,
- romfs — system plików dla urządzeń tylko do odczytu o małej pojemności,

- smbfs — sieciowy system plików firmy Microsoft, używany w systemach Windows,
- tmpfs — system plików przechowujący pliki w pamięci RAM,
- sysv — Coherent File System (systemy SCO, Xenix, SystemV, Coherent),
- udf — Univarsal Disk Format (płyty DVD),
- ufs — systemy BSD, SunOS, NeXTstep,
- umsdos — nakładka na system MS-DOS umożliwiająca użycie uniksowych praw dostępu,
- vfat — odmiana systemu FAT używana przez systemy Microsoft Windows (FAT12, FAT16, FAT32 z długimi nazwami plików).

Gwiazdką oznaczono systemy plików, których moduły nie są skompilowane w dystrybucji Red Hat 7.3. Aby z nich korzystać, należy przekompilować jądro systemu lub skompilować same moduły. Dla systemów plików ext2, iso9660 i tmpfs nie ma modułów, ponieważ są one na stałe wkompilowane w jądro systemu. Sposób obsługi niektórych spośród tych systemów opisany jest na stronie podręcznika systemowego polecenia `mount`; można też szukać informacji z plikach znajdujących się w katalogu */usr/src/linux/Documentation/filesystems*.

Jak już wcześniej wspomniano, nie ma jednego, uniwersalnego schematu partycjonowania dysku. Oprócz powszechnego schematu systemu DOS jądro Linuksa obsługuje także następujące schematy partycjonowania (gwiazdką oznaczono sterowniki nieobecne w jądrze dystrybucji Red Hat 7.3; aby z nich korzystać, należy zrekompilować jądro systemu):

- Acorn*,
- Alpha OS,
- Amiga*,
- Atari*,
- Macintosh,
- PC BIOS,
- BSD Disklabel,
- Minix,
- Solaris,
- Unixware,
- Dynamiczne dyski Windows 2000*,
- SGI,
- Ultrix*,
- Sun.

Nazwa Ultrix może wydawać się nieznajoma; schemat ten był najpierw używany w systemach DEC, następnie Compaq, a obecnie HP.

Gdy dokładnie przeszuka się Internet, można znaleźć jeszcze więcej modułów. Jak to bywa z otwartymi systemami operacyjnymi, każdy obdarzony odpowiednimi umiejętnościami programista może napisać własny moduł systemu plików.

Działanie dysku

Nasza definicja systemu plików zawiera pojęcie urządzenia przechowującego dane, przyjrzyjmy się więc najpowszechniej wykorzystywanemu urządzeniu — dyskowi twardemu. Dyskietki działają podobnie, z tą różnicą, że dyski są wymienne.

Z mechanicznego punktu widzenia napęd jest metalowym pudełkiem, zawierającym tzw. talerze pokryte materiałem ferromagnetycznym. Zwykle na tej samej osi znajduje się kilka talerzy, obracanych jednym silnikiem. Głowice odczytujące i zapisujące po każdej stronie każdego z talerzy są poruszane drugim silnikiem, który umieszcza je nad obszarem dysku zawierającym interesujące nas dane. Adres porcji danych na dysku określa się, podając numer cylindra (walcowata powierzchnia, obejmująca dane na wszystkich talerzach przy określonej pozycji głowic — numer cylindra określa więc pozycję głowic), głowicy (numer i stronę talerza) i sektora (porcja danych — domyślnie wielkości 512 bajtów — spośród danych dostępnych w podanym cylindrze dla podanej głowicy). Każdy dysk posiada elektroniczną kartę kontrolera, która w porozumieniu z kontrolerem dysku na płycie głównej komputera steruje położeniem głowic.

Zakres każdego z trzech elementów — adresy cylindrów, głowic i sektorów (*CHS* — *cylinders*, *heads*, *sectors*) — określa geometrię napędu. Jest ona zwykle wykrywana przez BIOS i przekazywana dalej do systemu operacyjnego. Linux może zignorować geometrię przekazaną przez BIOS i wykorzystać wartości przekazane jako argument jądra przy starcie systemu. Informacja o geometrii pomaga programowi ładującemu zlokalizować na dysku jądro systemu, a jądru pomaga zrozumieć organizację danych na dysku.

Jeżeli przyjrzymy się budowie tablicy partycji i trochę policzymy, możemy zauważyć, że na zakres wartości cylindra przeznaczone jest jedynie dziesięć bitów, więc żaden dysk nie może mieć więcej niż 1024 cylindry widziane przez BIOS (słynny limit 1024 cylindrów). Można sobie z tym poradzić, zmieniając nieco adresowanie dysku, czyli zwiększając liczbę głowic, dzięki czemu można użyć wystarczająco małej liczby cylindrów, aby zmieściła się w tablicy partycji. Ten sposób znany jest pod nazwą *LBA* (*logical block adressing* — logiczne adresowanie bloków). Nowoczesne układy BIOS (wyprodukowane około 1998 roku lub później) obsługują tę opcję, ale starsze nie. Jako że geometrię dysku można przekazać do jądra Linuksa przy starcie systemu, nie stanowi to dla użytkowników Linuksa takiego problemu, jak dla użytkowników innych systemów operacyjnych. Gdy wystąpią problemy z obsługą dużych dysków, dokładniejsze informacje na ten temat można znaleźć w dokumentach Large-Disk-HOWTO i Boot-Prompt-HOWTO.

Pierwszy sektor dysku jest najważniejszy, ponieważ zawiera tablicę partycji i kod programu ładującego. Kod ten jest wykonywany po inicjalizacji sprzętu przez BIOS. Przekazuje on sterowanie programowi ładującemu, który z kolei ładuje jądro systemu, oddając mu kontrolę nad komputerem. Dalsza inicjalizacja przebiega w znany już nam sposób.

Pierwszy sektor określany jest jako *MBR — Master Boot Record* (główny rekord startowy). Ma długość 512 bajtów: w pierwszych 446 znajduje się kod startowy, kolejne 64 bajty zawierają tablicę partycji, a ostatnie dwa — specjalny kod identyfikujący sektor jako tablicę partycji (szesnastkowe wartości 55 i AA). Bardziej szczegółowe informacje o MBR można znaleźć w rozdziale 11., „Kopie zapasowe, odtwarzanie danych, ratowanie systemu" [można przyjrzeć się szesnastkowej zawartości MBR, uruchamiając polecenie `fdisk` na danym dysku, wybierając `x` (dodatkowe funkcje), a następnie `d` (wypisanie zawartości tablicy partycji na ekran); polecenie `fdisk` zostanie niedługo dokładniej omówione].

Aby utworzyć na dysku system plików, najpierw należy utworzyć tablicę partycji. Kod startowy jest zapisany w następnej kolejności przez program ładujący — LILO, GRUB lub inny, właściwy dla uruchamianego systemu operacyjnego. Aby utworzyć system plików i przechowywać dane, nie jest potrzebny kod startowy; w zasadzie dyski zainstalowane inaczej niż jako */dev/hda* (*master* na pierwszym kablu) nie będą miały zapisanego kodu startowego — miejsce przeznaczone na kod najprawdopodobniej będzie puste. W dyskach SCSI kod startowy zostanie zapisany w dysku oznaczonym w BIOS-ie jako startowy.

Tablica partycji ma miejsce tylko na cztery partycje (gdy format był tworzony, musiano założyć, że cztery to całe mnóstwo). Aby obejść ten problem, jedna z tych czterech partycji (tradycyjnie czwarta) może zostać użyta jako partycja rozszerzona — zajmująca pozostałą część dysku i zawierająca własną tablicę partycji. Może ona zostać podzielona na maksymalnie 63 partycje w przypadku dysku IDE i 15 partycji w przypadku dysku SCSI.

Tablica partycji

Red Hat Linux 7.3 zawiera kilka narzędzi do tworzenia, przeglądania i modyfikacji tablicy partycji. Jedno z nich musiało zostać użyte podczas instalacji systemu. Wybór narzędzia do pracy z tablicą partycji jest kwestią gustu — niektórzy lubią tekstowy interfejs linii poleceń, podczas gdy inni wolą interfejs graficzny. Jako że niektóre z omawianych narzędzi mogą nie być zainstalowane w systemie, rozpoczniemy od najpowszechniej dostępnych, kończąc na graficznych narzędziach Red Hata.

fdisk

Odpowiednik linuksowego polecenia `fdisk` w systemie DOS służy do edycji tablicy partycji, tworzenia podstawowej struktury dysku (tzw. format niskiego poziomu — *low-level format*) i zapisania kodu startowego w sektorze MBR. Linuksowy `fdisk`, utworzony w uniksowej tradycji programów wykonujących jedno zadanie, ale za to dobrze, służy jedynie do edytowania tablicy partycji. Ma to znaczenie przy odzyskiwaniu dysku, omawianym w rozdziale 11.

Program *fdisk* należy uruchamiać jako superużytkownik (*root*). Obsługuje on tylko dyski twarde (IDE i SCSI); urządzenie przekazuje się jako argument. Dyski USB dostępne są za pośrednictwem emulacji SCSI i traktowane jak urządzenia SCSI. Otwórzmy za pomocą programu *fdisk* tablicę partycji pierwszego dysku IDE w systemie[1]:

```
# fdisk /dev/hda

The number of cylinders for this disk is set to 4111.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

Polecenie `m` wyświetla tekst pomocy[2]:

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Polecenie `p` wyświetla zawartość tablicy partycji:

```
Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 4111 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1       653   5245191    c  Win95 FAT32 (LBA)
/dev/hda2           654      2612  15735667+  83  Linux
/dev/hda3          2613      2939   2626627+  a5  FreeBSD
/dev/hda4          2940      4111   9414090    5  Extended
```

[1] Ilość cylindrów tego dysku jest ustawiona na 4111. Nie ma w tym nic złego, ale jest to więcej niż 1024, i może w pewnych przypadkach sprawiać kłopoty z: 1) oprogramowaniem uruchamianym przy starcie (np. stare wersje LILO); 2) uruchamianiem innych systemów i partycjonowaniem dysku przy ich użyciu (np. FDISK systemu DOS lub OS/2).

[2] a — przełącz flagę „partycja startowa”; b — edytuj tablicę BSD; c — przełącz flagę zgodności z systemem DOS; d — usuń partycję; l — pokaż znane typy partycji; m — wyświetl to menu; n — utwórz partycję; o — utwórz nową, pustą tablicę partycji DOS; p — pokaż tablicę partycji; q — wyjdź z programu, nie zachowując zmian; s — utwórz nową, pustą tablisę partycji Sun; t — zmień typ partycji; u — zmień używane jednostki rozmiaru; v — sprawdź poprawność tablicy partycji; w — zapisz tablicę partycji na dysk i wyjdź z programu; x — dodatkowe funkcje (tylko dla ekspertów)

```
/dev/hda5          2940       2952     104391   82  Linux swap
/dev/hda6          2953       3475    4200966   83  Linux
/dev/hda7          3476       4111    5108638+  83  Linux
```

Starsze wersje programu *fdisk* domyślnie otwierały dysk */dev/hda*, ale autor zdecydował, że nie było to dobrym pomysłem, więc teraz trzeba zawsze podać nazwę urządzenia.

Zabawa programem *fdisk* jest niebezpieczna tylko wtedy, gdy na koniec pracy zapisze się zmiany w tablicy partycji. Ponieważ należy to bezpośrednio nakazać, można z czystym sumieniem uruchomić program *fdisk* i wprowadzać dowolne zmiany, pamiętając, żeby nie wychodzić za pomocą polecenia `w`, tylko `q`. Uzbrojony w tę wiedzę możesz spokojnie bawić się programem; żeby jednak nie mieć najmniejszej szansy zepsucia czegokolwiek, możesz użyć opcji `-l` (litery „L”, nie cyfry „1”):

```
# fdisk -l /dev/hda
```

Wtedy *fdisk* tylko wyświetli zawartość tablicy partycji, nie wchodząc w tryb edycji. Aby zachować kopię tablicy partycji (co zwykle jest dobrym pomysłem), można przekierować wyjście polecenia do pliku:

```
# fdisk -l /dev/hda >tablicapartycji.txt
```

lub wysłać je bezpośrednio do drukarki:

```
# fdisk -l |lpr
```

W pierwszym przykładzie użyliśmy przekierowania powłoki `>`, aby skierować strumień wyjściowy polecenia do pliku; w drugim użyliśmy potoku `|`, podając wyjście programu *fdisk* bezpośrednio na wejście programu *lpr* (zakładając, że mamy w systemie poprawnie skonfigurowaną drukarkę).

sfdisk

Działający z linii poleceń program *sfdisk* jest opisany na swojej stronie podręcznika systemowego jako narzędzie o ogromnych możliwościach tylko dla hackerów. Jest to najprawdopodobniej przykład poczucia humoru programistów, ponieważ *sfdisk* ma wbudowany szereg zabezpieczeń chroniących przed wypadkami przy pracy. Składnia poleceń jest inna niż w programie *fdisk*, ale prosta i służy zasadniczo do tego samego.

Program *sfdisk* może sprawiać nieco dziwaczne wrażenie podczas używania. Poniżej przedstawiony jest tekst pomocy[3]:

[3] Sposób użycia: `sfdisk [opcje] urządzenie...` (urządzenie: coś w stylu */dev/hda* lub */dev/sda*). Przydatne opcje: `-s` (albo `--show-size`) — pokaż rozmiary partycji; `-c` — pokaż lub zmień typ partycji; `-l` — pokaż listę partycji każdego z urządzeń; `-d` — jak wyżej, ale w formacie odpowiednim do późniejszego przekazania do programu; `-i` — numeracja cylindrów itp. od 1, a nie od 0; `-uS`, `-uB`, `-uC`, `-uM` — użyj sektorów/bloków/cylindrów/megabajtów jako jednostek rozmiaru; `-T` — wyświetl znane tablice partycji; `-D` — zmarnuj trochę przestrzeni w celu uzyskania zgodności z systemem DOS; `-R` — zmuś jądro systemu do ponownego odczytania tablicy partycji; `-N#` — zmień tylko partycję o numerze #; `-n` — nie zapisuj nic na dysk; `-O plik` — zapisz nadpisywane sektory do pliku; `-I plik` — przywróć nadpisane wcześniej sektory z pliku; `-V` — wypisz wersję; `-?` — wypisz tę wiadomość. Niebezpieczne opcje: `-g` — pokaż geometrię dysku (według jądra systemu); `-x` — obsłuż także partycje rozszerzone; `-L` — nie narzekaj na rzeczy bez znaczenia dla Linuksa; `-q` — wstrzymaj ostrzeżenia. Można także wyłączyć kontrolę spójności opcją `-f` — rób, co mówię, nawet jeśli to jest głupie.

```
# sfdisk
sfdisk version 3.07 (aeb@cwi.nl, 990908)
Usage: sfdisk [options] device ...
device: something like /dev/hda or /dev/sda
useful options:
    -s [or --show-size]: list size of a partition
    -c [or --id]:        print or change partition Id
    -l [or --list]:      list partitions of each device
    -d [or --dump]:      idem, but in a format suitable for later input
    -i [or --increment]: number cylinders etc. from 1 instead of from 0
    -uS, -uB, -uC, -uM:  accept/report in units of sectors/blocks/cylinders/MB
    -T [or --list-types]:list the known partition types
    -D [or --DOS]:       for DOS-compatibility: waste a little space
    -R [or --re-read]:   make kernel reread partition table
    -N# :                change only the partition with number #
    -n :                 do not actually write to disk
    -O file :            save the sectors that will be overwritten to file
    -I file :            restore these sectors again
    -v [or --version]:   print version
    -? [or --help]:      print this message
dangerous options:
    -g [or --show-geometry]: print the kernel's idea of the geometry
    -x [or --show-extended]: also list extended partitions on output
                             or expect descriptors for them on input
    -L  [or --Linux]:      do not complain about things irrelevant for Linux
    -q  [or --quiet]:      suppress warning messages
    You can override the detected geometry using:
    -C# [or --cylinders #]:set the number of cylinders to use
    -H# [or --heads #]:    set the number of heads to use
    -S# [or --sectors #]:  set the number of sectors to use
You can disable all consistency checking with:
    -f  [or --force]:      do what I say, even if it is stupid
```

Podanie nazwy urządzenia jako parametru zaowocuje wypisaniem na ekran ostrzeżenia:

```
# sfdisk /dev/hda
Checking that no-one is using this disk right now ...
BLKRRPART: Urządzenie lub zasoby są zajęte

This disk is currently in use - repartitioning is probably a bad idea.
Umount all file systems, and swapoff all swap partitions on this disk.
Use the --no-reread flag to suppress this check.
Use the --force flag to overrule all checks.
```

Nawet podanie nazwy urządzenia jest traktowane jako podejrzane; należy przekazać opcję `-f`, aby zmusić program do otwarcia tablicy partycji. Podczas dalszej pracy można spodziewać się kolejnych ostrzeżeń. Aby popełnić błąd, należy się bardzo postarać.

```
# sfdisk --force /dev/hda
Checking that no-one is using this disk right now ...
BLKRRPART: Urządzenie lub zasoby są zajęte

This disk is currently in use - repartitioning is probably a bad idea.
Umount all file systems, and swapoff all swap partitions on this disk.
Use the --no-reread flag to suppress this check.

Disk /dev/hda: 4111 cylinders, 255 heads, 63 sectors/track
Old situation:
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0
```

```
   Device Boot Start     End   #cyls   #blocks   Id  System
/dev/hda1   *      0+     652     653-   5245191    c  Win95 FAT32 (LBA)
/dev/hda2        653    2611    1959  15735667+  83  Linux
/dev/hda3       2612    2938     327   2626627+  a5  FreeBSD
/dev/hda4       2939    4110    1172   9414090    5  Extended
/dev/hda5       2939+   2951      13-   104391   82  Linux swap
/dev/hda6       2952+   3474     523-  4200966   83  Linux
/dev/hda7       3475+   4110     636-  5108638+  83  Linux
/dev/hda8       2624+   2938     315-  2524227+
/dev/hda9       2612    2624-     13-   102400
Input in the following format; absent fields get a default value.
<start> <size> <type [E,S,L,X,hex]> <bootable [-,*]> <c,h,s> <c,h,s>
Usually you only need to specify <start> and <size> (and perhaps <type>).

/dev/hda1 :
```

Największą przewagą polecenia `sfdisk` nad programem *fdisk* jest możliwość pisania skryptów automatyzujących jego pracę (program nie wyświetla wtedy wszystkich ostrzeżeń). Aby korzystać z programu w sposób interaktywny, należy podać opcję `-f` z linii poleceń, jak w poprzednim przykładzie.

Program zawiera kilka przydatnych zabezpieczeń używanych przy manipulacji tablicą partycji; użycie go może być dobrym wyborem dla wieloużytkownikowego systemu sieciowego.

GNUparted

Starsze wersje dystrybucji Red Hat korzystały podczas instalacji z programu partycjonującego Disk Druid; starsi użytkownicy Red Hata z pewnością go używali. Kod programu Disk Druid został zastąpiony programem GNUparted (znanym też, od polecenia uruchamiającego program, jako *parted*) — edytorem partycji GNU, programem o bardzo dużych możliwościach. Aktualnie używany podczas instalacji interfejs graficzny wygląda tak samo jak starsze wersje, ale jest tylko nakładką na program *parted*, działający z linii poleceń. Może on tworzyć, kasować, przenosić, kopiować, a nawet zmieniać rozmiar partycji ext2 i FAT32. Może być użyty interaktywnie lub za pośrednictwem skryptu. Strona podręcznika systemowego lub systemu info zawiera dokładniejsze informacje na temat sposobu użycia tego programu.

Pliki

Jeżeli miałbyś z tego rozdziału nauczyć się tylko jednej rzeczy, powinieneś nauczyć się tego, że wszystko jest plikiem. To jest główna idea uniksowych systemów plików.

Istnieje kilka różnych rodzajów plików; tutaj zajmiemy się głównie dwoma z nich: specjalnymi plikami znakowymi (*character special files*) i specjalnymi plikami blokowymi (*block special files*). Służą one odpowiednio do obsługi urządzeń znakowych i blokowych; używa się ich do zapisywania danych do urządzenia i odczytu z urządzenia. Dla informacji — pozostałymi rodzajami są zwykłe pliki, kolejki FIFO (*FIFOs, named pipes*), katalogi, dowiązania symboliczne (*symbolic links*) i gniazdka (*UNIX domain sockets*).

W systemie z zainstalowaną dokumentacją jądra znajduje się plik */usr/src/linux/devices.txt*. Oto fragment tego pliku:

```
3 char        Pseudo-TTY slaves
                0 = /dev/ttyp0        First PTY slave
                1 = /dev/ttyp1        Second PTY slave
                  ...
              255 = /dev/ttyef        256th PTY slave

              These are the old-style (BSD) PTY devices; Unix98
              devices are on major 136 and above.

  block       First MFM, RLL and IDE hard disk/CD-ROM interface
                0 = /dev/hda          Master: whole disk (or CD-ROM)
               64 = /dev/hdb          Slave: whole disk (or CD-ROM)

              For partitions, add to the whole disk device number:
                0 = /dev/hd?          Whole disk
                1 = /dev/hd?1         First partition
                2 = /dev/hd?2         Second partition
                  ...
               63 = /dev/hd?63        63rd partition

              For Linux/i386, partitions 1-4 are the primary
              partitions, and 5 and above are logical partitions.
              Other versions of Linux use partitioning schemes
              appropriate to their respective architectures.
```

Numer 3 w lewym górnym rogu jest głównym numerem urządzenia (*major device number*), oznaczającym rodzaj urządzenia. W powyższym fragmencie odnosi się on zarówno do urządzeń znakowych (pseudoterminali), jak i blokowych (dysków IDE). Kolumny liczb poniżej typu urządzenia są pobocznym numerem urządzenia (*minor device number*); każde urządzenie można jednoznacznie zidentyfikować, podając numer główny i poboczny. W komunikatach błędów czasem są podane jedynie numery urządzenia. Może to być irytujące, jeżeli nie wie się o istnieniu pliku *devices.txt*, pomagającego ustalić znaczenie numerów. Warto przejrzeć ten plik — zawiera on odpowiedzi na wiele często spotykanych problemów.

Wszystkie pliki w katalogu */dev* można przejrzeć, wydając polecenie:

```
# ls -l --sort=none /dev | less
```

Opcja `--sort=none` powoduje, że pliki pojawiają się w grupach odpowiadających numerom urządzeń; umieszczenie w potoku polecenia `less` pozwala przeglądać listę przy użyciu klawiszy *PageUp* i *PageDown*.

System plików DEVFS

Niedługo w powszechnym użyciu będzie nowy, alternatywny system plików urządzeń, devfs (*Device Filesystem*). Wyświetlane w nim są tylko urządzenia faktycznie obecne w systemie, w innym schemacie nazewnictwa niż dotychczasowy. Może być używany równolegle z systemem tradycyjnym.

Na przykład plik w starym systemie nazywający się */dev/hda1*, w nowym będzie się nazywał */dev/ide/hd/c0b0t0u0p1* i będzie dowiązaniem symbolicznym do */dev/ide/host0/bus0/target0/lun0/part1*. Może się to wydawać bardziej skomplikowane, ale głównie ze względu na nieznajomość tego systemu.

Devfs jest w pewnym stopniu podobny do systemu plików */proc* — też jest systemem wirtualnym; pliki są tworzone na bieżąco przez kod jądra, nie zajmując miejsca na dysku. Jaka jest główna przewaga tego systemu nad tradycyjnym? Ograniczenia wartości numerów urządzeń (do 128), konieczność utrzymywania pliku *devices.txt*, aby nowe sterowniki nie kolidowały ze sobą (można by zwiększyć zakres numerów urządzeń, ale stworzyłoby to problemy z wydajnością) i konieczność umieszczania plików urządzeń na głównej partycji, praktycznie rzecz biorąc, uniemożliwiająca zamontowanie głównej partycji tylko do odczytu, uruchomienie systemu z nieuniksowego systemu plików lub z pamięci ROM lub współdzielenie głównego katalogu jako dysku sieciowego NFS — to tylko główne niedogodności systemu tradycyjnego. Devfs rozwiązuje wszystkie te problemy i czyni Linuksa jeszcze bardziej elastycznym. Podobnych schematów używają już systemy FreeBSD, Beos, Plan9 i QNX — jest to nowość tylko dla Linuksa.

Dalsze szczegóły można znaleźć w FAQ dotyczącym systemu devfs pod adresem *http://www.atnf.csiro.au/~rgooch/linux/docs/devfs.html*.

Nazewnictwo urządzeń blokowych i znakowych

Tradycyjny schemat nazewnictwa urządzeń blokowych jest używany już od dłuższego czasu. Jak widać z przytoczonego przed chwilą fragmentu pliku *devices.txt*, nazwą pierwszej partycji pierwszego dysku IDE byłoby */dev/hda1*. Dla dysku SCSI byłoby to */dev/sda1* (wymienione w innym miejscu *devices.txt*). Wszystkie nazwy są opisane w pliku *devices.txt*, a wszystkie urządzenia utworzone w systemie znajdują się w katalogu */dev*. Niektóre z plików w tym katalogu są dowiązaniami symbolicznymi, na przykład */dev/cdrom* jest zwykle tylko dowiązaniem do odpowiedniego urządzenia (*/dev/hdc* może */dev/scd0*?); innymi często używanymi dowiązaniami są */dev/mouse* i */dev/modem*.

Tworzenie urządzeń przy użyciu polecenia mknod

Nietrudno zauważyć w katalogu */dev* pliki urządzeń niepodłączonych do komputera. Aby system mógł korzystać z urządzenia, plik musi istnieć — system nie jest w stanie tworzyć plików urządzeń na bieżąco, dystrybucja zawiera więc sporą ilość potencjalnie potrzebnych plików urządzeń. Jeżeli trzeba użyć pliku nieobecnego w */dev*, do jego tworzenia służy polecenie `mknod`. Tworzenie urządzenia jest proste, o ile tylko zna się typ urządzenia oraz jego numer główny i poboczny. Oto składnia polecenia:

```
# mknod [opcja]… nazwa typ numer_główny numer_poboczny
```

Przydatna jest opcja `-m`, pozwalająca ustalić prawa dostępu do pliku (jak dla `chmod`) od razu przy jego tworzeniu, a nie oddzielnie.

Nazwę, typ i numery pliku można znaleźć w pliku *devices.txt* — jest w nim nawet seria numerów przeznaczonych do eksperymentów z własnymi sterownikami jądra.

Urządzenia znakowe, blokowe i specjalne

Urządzenie znakowe jest plikiem przetwarzającym dane jako strumień znaków, bajt po bajcie. Przykładami urządzeń znakowych są terminale, napędy taśm magnetycznych, klawiatura, karty dźwiękowe, sieciowy system plików Coda i inne.

Urządzenia blokowe posiadają początek, koniec i ustalony rozmiar; dane można zapisać na dowolnym miejscu urządzenia i odczytać je z niego dowolnego miejsca urządzenia. Jako że urządzenia blokowe mogą być dużo większe niż objętość danych na nich zawartych, specjalizowane programy, jak *tar* czy *cpio*, operują raczej na zawartości plików niż ich rozmiarze, mogąc zachowywać i odczytywać pliki bezpośrednio z urządzenia, nie wymagając sformatowanego pliku na urządzeniu blokowym. Działa to szczególnie dobrze z napędami taśm magnetycznych, ponieważ są one raczej urządzeniami znakowymi niż blokowymi i nie są sformatowane w sposób właściwy dla urządzeń blokowych.

Niżej wymieniamy niektóre z ważniejszych urządzeń specjalnych.

- */dev/null* — puste urządzenie, określane czasem jako „wiadro na bity” (*bit bucket*). Wszystko, co zapiszemy do tego urządzenia jest ignorowane. Jest przydatne do przekierowania informacji, gdy nie chce się ich wyświetlać ani przekierowywać do pliku.
- */dev/zero* — to urządzenie dostarcza niewyczerpanego strumienia zer. Jest przydatne do nadpisywania plików lub urządzeń zerami.

Systemy plików aktualnie dostępne w systemie

Najczęściej pracujemy z lokalnymi systemami plików, umieszczonymi na używanym komputerze. Jakie są w danej chwili dostępne? Różni się to zależnie od ustawień jądra systemu i sprzętu; sprawdzamy, zaglądając do pliku */proc/filesystems*:

```
# cat /proc/filesystems
nodev   rootfs
nodev   bdev
nodev   proc
nodev   sockfs
nodev   tmpfs
nodev   shm
nodev   pipefs
        ext2
nodev   ramfs
        iso9660
nodev   devpts
        ext3
```

```
nodev   usbdevfs
nodev   usbfs
        vfat
```

Interesujące dla nas są wpisy niepoprzedzone słowem `nodev`.

Systemy plików obsługiwane przez Linuksa

Systemy plików możemy podzielić na dwa rodzaje: sieciowe i dyskowe, omówimy je w następnych podrozdziałach.

Mimo że Linux obsługuje szyfrowane systemy plików, nie są one obsługiwane przez dystrybucję Red Hata. Dokładniejsze informacje na temat własnoręcznej konfiguracji można znaleźć pod adresem *http://www.linuxdoc.org/HOWTO/Loopback-Encrypted-Filesystem-HOWTO.html*.

Sieciowe systemy plików

Sieciowe systemy plików fizycznie znajdują się w odległym systemie, ale są widoczne jakby były lokalnie zamontowanym dyskiem. Najpowszechniej stosowane typy wyliczamy poniżej.

- NFS — powszechnie używany sieciowy system plików opracowany przez firmę Sun. Nie posiada wbudowanych mechanizmów zabezpieczeń, ponieważ został zaprojektowany do pracy w przyjaznej sieci. Jest uważany przez niektórych za problematyczny, jednak jest łatwy w implementacji. Z reguły używany do współdzielenia plików między systemami uniksowymi.
- SMB — opierający się na sieci protokół *System Message Block* opracowany przez firmę Microsoft. Linuksowa implementacja protokołu znana jest pod nazwą Samba i działa całkiem dobrze, dopóki Microsoft nie wprowadzi kolejnych zmian do protokołu. Z reguły jest używany do komunikacji między systemami Windows i Linux.

Dyskowe systemy plików

Dyskowe systemy plików znajdują się na urządzeniu fizycznym, na przykład dysku twardym, bezpośrednio podłączonym do lokalnego komputera.

- FAT — nastawiony na dysk, oparty na tablicach (strukturach listy jednokierunkowej) system plików Microsoftu. Jest on regularnie rozszerzany w celu dodania funkcjonalności. Profesjonalnym systemem plików Microsoftu jest NTFS.
- ext2, ext3 i ReiserFS są opartymi na i-węzłach uniksowymi systemami plików.

Bliższe spojrzenie na system plików ext2

Ostatnimi czasy standardowym systemem plików Linuksa jest ext2, chociaż używano i używa się także innych systemów plików. Dla użytkowników dystrybucji Red Hat aktualną technologią jest system ext3. Mają one wiele wspólnych cech, więc najpierw omówimy dokładnie ext2, a następnie ext3.

Charakterystyka systemu ext2

Pierwotny *Extended File System* został nazwany „ext"; druga jego wersja nazywa się, o dziwo, ext2. Ten linuksowy system plików jest wciąż rozwijany — nietrudno się domyśleć, skąd wzięła się nazwa systemu ext3. System ext2 może obsłużyć pliki o rozmiarze do 2 GB, katalogi o rozmiarze 2 TB i pliki o nazwach długości 255 znaków (jeżeli limit ten jest niewystarczający, po wprowadzeniu specjalnej „łaty" na jądro systemu nazwy plików mogą mieć do 1024 znaków). Jednym z największych usprawnień systemu ext2 była możliwość wydajnej alokacji i wykorzystania wolnej przestrzeni. Jest ono tak wydajne, że systemy plików ext2zwykle nie potrzebują defragmentacji (jest to odpowiedź na często zadawane pytanie: tak, istnieje program do defragmentacji dysków dla Linuksa, ale jest bardzo rzadko używany, nie jest obecny w większości dystrybucji — w tym w Red Hat Linuksie — i jego użycie nie jest zalecane). Dynamiczna alokacja zasobów jest także piętą Achillesa systemu ext2 — gdy plik jest kasowany, jego i-węzeł jest usuwany, a bloki dysku składające się na plik są uwalniane i mogą być od razu ponownie zaalokowane, a skasowane dane — bezpowrotnie utracone. Istnieją programy „odkasowujące" pliki dla systemu ext2, ale ich skuteczność w intensywnie używanych, wieloużytkownikowych systemach bywa znikoma; w systemie domowym szanse odzyskania danych są wiele większe, ale nie jest to gwarantowane ze strony systemu.

Polecenie sync

Linux buforuje zapisy do urządzeń, dlatego zapisane dane nie pojawią się na dysku dopóki nie zapełni się bufor, jądro systemu każe wykonać zapis lub my polecamy go wykonać, używając polecenia `sync`; tradycyjnie polecenia używa się dwukrotnie:

```
# sync ; sync
```

Podwójne wywołanie polecenia jest całkowicie zbędne.

Struktura systemu ext2

Każdy system plików ma swoją charakterystyczną strukturę. Różne struktury używane są do innych celów związanych z wydajnością, bezpieczeństwem, a w przypadku zastrzeżonych projektów — nawet zmniejszeniem zgodności z innymi systemami. System ext2 został zaprojektowany dla zgodności z zasadami systemów uniksowych, w szczególności zasady „wszystko jest plikiem". Tak więc na przykład katalog w systemie plików ext2 jest plikiem zawierającym listę nazw plików znajdujących się w tym katalogu oraz położenie tych plików na dysku; nazwy przechowywane są w strukturze listy jednokierunkowej, co pozwala uniknąć marnowania przestrzeni dyskowej związanego z różnymi długościami nazw plików.

I-węzły

Plik w systemie ext2 zaczyna się od i-węzła (*inode*), zawierającego opis pliku: jego typ, prawa dostępu, właściciela i grupę, znaczniki czasu i wskaźniki do bloków danych z zawartością pliku. Podczas korzystania z pliku jądro systemu używa tych wskaźników do znalezienia konkretnego miejsca na dysku zawierającego dane.

Dowiązania twarde i symboliczne

W związku ze strukturą systemu plików ext2 pojedynczemu plikowi można przypisać kilka nazw (wpisów w katalogu); są to tak zwane dowiązania twarde (*hard links*). Można także utworzyć specjalny plik, dowiązanie symboliczne (*symbolic link*), odwołujący się do nazwy innego pliku. Rodzaje te różnią się tym, że dowiązanie twarde odnosi się do samych danych (wskazuje na i-węzeł), a symboliczne — do nazwy pliku. Oba rodzaje dowiązań widzimy w spisie plików jako zwykły plik i można operować zarówno na właściwym pliku, jak i na dowiązaniu. Głównym zyskiem z posługiwania się dowiązaniami jest ograniczenie zużycia przestrzeni dyskowej potrzebnej do przechowywania zduplikowanych plików.

Fizyczna struktura dysku

Gdybyśmy mieli obrazowo przedstawić strukturę danych na fizycznym dysku w systemie plików ext2, przypominałyby one serię pudełek — bloków. Pierwszy blok dysku jest blokiem specjalnym, zawierającym sektor startowy; każdy z następnych bloków zawiera system operacyjny, aplikacje bądź dane.

Każdy blok złożony jest z mniejszych fragmentów danych: superbloku (nazywanego w ten sposób, ponieważ zawiera powieloną informację o danym systemie plików), powielonych deskryptorów systemu plików, mapy bitowej bloku, mapy bitowej tablicy i-węzłów, zawartości tablicy i-węzłów, a następnie bloków danych. Powielona informacja zwiększa niezawodność systemu i ułatwia ratowanie zawartości dysku po awariach i błędach.

Jak duże są te bloki? Domyślnym rozmiarem są 1024 bajty, ale rozmiar można powiększyć lub zmniejszyć podczas tworzenia systemu plików. Optymalny rozmiar zależy od zastosowania systemu — do przechowywania dużych plików lepiej użyć dużych bloków, aby przyspieszyć działanie dysku kosztem zmarnowanej przestrzeni dla małych plików; do przechowywania małych plików bardziej sensowne jest ustalenie mniejszego rozmiaru bloku. Niektóre z systemów mogą wymagać pewnego czasu monitorowania i pomiarów prędkości w celu ustalenia optymalnej wartości. Rozmiar bloków danych można ustalić podczas formatowania partycji.

Sprawdzanie integralności danych programem fsck

Spójność systemu plików zapewnia użycie polecenia `fsck`, jednego z pięciu programów biblioteki ext2 używanych do utrzymywania i modyfikacji systemu plików ext2.

Stan systemu plików jest w systemie ext2 śledzony. Po zamontowaniu systemu plików z możliwością zapisu jądro zaznacza w superbloku odpowiednie pole, mówiące, że system jest nieczysty; po poprawnym wymontowaniu system jest oznaczany jako czysty. Jeżeli system plików nie został poprawnie rozmontowany, może zawierać uszkodzone dane, ponieważ część danych mogła nie zostać zapisana (ten problem próbują wyeliminować systemy plików z żurnalem, jak chociażby ext3). Podczas startu systemu flaga ta jest sprawdzana i jeżeli system plików jest nieczysty, uruchamiany jest program *fsck* (w zasadzie *fsck* jest programem sprawdzającym rodzaj systemu plików na danej partycji i uruchamiającym program odpowiedni dla danego systemu: *fsck.minix*, *fsck.ext2*, *fsck.ext3*, *fsck.reiserfs*, *fsck.msdos* lub *fsck.vfat*). Jeżeli jądro systemu wykryje błąd w strukturze systemu plików lub w superbloku, system oznaczany jest jako błędny, co forsuje sprawdzenie systemu plików nawet, gdy nic poza tym nie wskazuje na potrzebę uruchomienia *fsck*.

Domyślnie system uruchomi *fsck* na danym systemie plików po zadanej liczbie restartów systemu niezależnie od stanu pola czystości, zależnie od licznika zamontowań systemu (przechowywanego w superbloku) lub po upływie określonego czasu od ostatniego sprawdzenia (też zapisanego w superbloku). Parametry te można dostosowywać za pomocą polecenia `tune2fs`; umożliwia ono także zmianę sposobu obsługi błędu w systemie plików przez jądro a także ilość bloków zarezerwowanych dla superużytkownika. Ta ostatnia opcja jest przydatna na wyjątkowo dużych lub szczególnie małych dyskach — można wtedy udostępnić użytkownikom więcej przestrzeni dyskowej.

Sposób działania fsck

Podczas działania *fsck* przetwarza na różne sposoby informację znalezioną w systemie plików. Gdy znajdzie katalog, niezwiązany z głównym drzewem katalogów lub nieskasowany plik bez wpisów w żadnym katalogu, przenosi plik lub katalog do katalogu */lost+found*, tworzonego na każdej partycji podczas formatowania. Pewna ilość bloków w systemie plików jest zarezerwowana do tego i innych celów dla superużytkownika. Można zmniejszyć ten przydział, dając więcej miejsca do wykorzystania użytkownikom, przekazując odpowiednie argumenty programowi *mke2fs* podczas tworzenia systemu plików.

Przeglądanie systemu plików ext2

Red Hat Linux 7.3 zawiera polecenie `dumpe2fs`, umożliwiające obejrzenie struktury systemu plików ext2. Przykład działania tego programu znajduje się w dalszej części tego rozdziału.

Składnia polecenia jest następująca:

```
dumpe2fs [-bfhixV] [-ob superblok] [-oB rozmiar_bloku] urządzenie
```

Dwie najbardziej przydatne opcje działają, według strony podręcznika systemowego, następująco:

- `-b` — wypisuje bloki oznaczone w systemie plików jako błędne,
- `-f` — zmusza `dumpe2fs` do wyświetlenia pliku, nawet jeżeli ma on ustawione flagi, których program nie rozumie (co może sprawić, że część wyświetlonych informacji będzie pozbawiona znaczenia).

Kolejne dwie opcje przydają się podczas odzyskiwania utraconych danych. Zdesperowany i posunięty do ostateczności administrator systemu (czyli czytelnik) może okazać się wymienionym czarodziejem:

- `-ob` *superblok* — użyj podczas przeglądania systemu innego superbloku; nie jest to zwykle potrzebne komukolwiek poza czarodziejem przyglądającym się szczątkom bardzo nieprzyjemnie uszkodzonego systemu plików;
- `-oB` *rozmiar_bloku* — przeglądając system, używaj bloków o podanym rozmiarze; nie jest to zwykle potrzebne komukolwiek poza czarodziejem próbującym coś odzyskać ze szczątek uszkodzonego systemu plików.

Do manipulowania systemem ext2 a także do naprawy uszkodzonego systemu plików lub do odzyskiwania danych można też użyć programu *debugfs*. Jako narzędzie o dużych możliwościach, domyślnie uruchamia się bez możliwości zapisania jakichkolwiek zmian do systemu plików; zapis należy uaktywnić podczas uruchamiania polecenia. Wydając polecenie `help` w linii poleceń programu, zobaczymy spis dostępnych poleceń.

Alternatywne systemy plików Linuksa

Ze względu na budowę wirtualnego systemu plików VFS, w systemie Linux można łatwo korzystać z alternatywnych systemów plików.

Ostatnimi czasy bardzo aktywnie tworzone są systemy plików z tzw. żurnalem (*journaling filesystems*). W takich systemach przed wprowadzeniem jakichkolwiek zmian w faktycznym systemie plików w tzw. żurnalu zapisywana jest porcja metadanych — opis wprowadzanych zmian. Dzięki temu w razie awarii systemu (spowodowanej chociażby przerwą w dostawie prądu) zmiany w plikach zostaną wprowadzone albo w całości, albo wcale — albo w żurnalu znajduje się opis wprowadzanych zmian umożliwiający dokończenie potencjalnie przerwanej w połowie operacji zapisu, albo opisu tam nie ma i wtedy zmiana zostanie w całości porzucona. Stosowane są różne implementacje tego pomysłu, używające odmiennych kombinacji danych i metadanych, ogólnie rzecz biorąc, celem jest doprowadzenie systemu plików podczas późniejszego startu systemu do stanu, w jakim znalazłby się bez niespodziewanej zapaści systemu. Zawsze istnieje rozdźwięk między wydajnością a ilością danych zapisanych w żurnalu; system plików ext3, omawiany w dalszej części tego rozdziału, daje administratorowi możliwość ustawienia opcji definiujących rozsądny dla danego systemu kompromis. Głównym problemem w systemie plików ext2 jest czas sprawdzania systemu plików po awarii systemu, który może trwać długo nawet na systemach umiarkowanej wielkości; w dużych systemach `fsck` może być powodem nawet kilkugodzinnej niedostępności systemu.

Komercyjne systemy uniksowe korzystają z różnych rodzajów systemów plików z żurnalem; na Linuksa przenoszony jest system XFS z SGI, a także JFS firmy IBM (oparty na implemetacji JFS w systemie OS/2, nie w systemie AIX). Kontynuowane są prace

nad systemem ext3, ale w chwili obecnej najbardziej zaawansowany jest system plików ReiserFS. Mimo że ext3 jest wystarczająco stabilny, aby być domyślnym systemem plików w Red Hat Linuksie 7.3 i mieć zastosowanie w systemach produkcyjnych Red Hata i innych firm, jest on razem z innymi systemami z żurnalem wciąż uważany za oprogramowanie beta, nie nadające się do użytku w krytycznych systemach. Dzisiaj ext2 pozostaje właściwym systemem plików Linuksa, chociaż spodziewamy się, że nowy standard wyłoni się jeszcze w 2002 roku. Z obsługą Red Hata dla ext3 jest on bardzo prawdopodobnym kandydatem, ale bogactwo dostępnych dla Linuksa systemów plików jest bardzo duże.

Ext3

Red Hat wybrał obsługę ext3 jako domyślnego systemu plików z żurnalem w swojej dystrybucje. Inne dystrybucje, jak SuSE czy Mandrake, obsługują system ReiserFS. Na dziś istnieje ograniczona możliwość korzystania z systemów XFS i JFS, ale najprawdopodobniej służyć one będą głównie użytkownikom tych systemów plików do korzystania ze sformatowanych już dysków pod Linuksem.

Uzasadnienie wyboru ext3 przez Red Hata jest nie do odparcia: system ten zapewnia dostępność, spójność danych i szybkość działania podobną do innych systemów plików z żurnalem, ale ma jedną unikalną zaletę — łatwe przejście z systemu ext2 na ext3 w sposób niepodatny na pomyłki użytkownika.

Można wybrać korzystanie z systemu plików ext3 już przy instalacji; można też przekonwertować po uaktualnieniu systemu do wersji 7.3 istniejącą partycję ext2 do systemu ext3.

Konwersja systemu plików ext2 na ext3

Aby dodać możliwość korzystania z żurnala do istniejącego systemu plików ext2, należy użyć polecenia `tune2fs`. Aby na przykład przetworzyć sformatowaną już partycję */dev/hda2*, wydajemy polecenie:

```
# tune2fs -j /dev/hda2
```

Nie ma znaczenia, czy podczas tej operacji zmieniana partycja jest zamontowana, czyli; jedyną różnicą jest pojawienie się nowego pliku *.journal* w katalogu, w którym zamontowano daną partycję — o ile była ona wcześniej zamontowana.

Następnie znajdźmy odpowiednią linijkę w pliku */etc/fstab* i zmieńmy zapis ext2 na ext3. Po następnym uruchomieniu systemu zostanie ona zamontowana jako ext3.

Jeżeli zmiana dotyczyła głównej partycji, nie będzie można jej wymontować przed wydaniem polecenia `tune2fs`, ale nie ma to znaczenia. Po utworzeniu żurnalu pojawi się tylko plik *.journal*. Należy jednak wtedy utworzyć plik *initrd* (RAM-dysk startowy, zawierający jądro Linuksa i minimalny system, umożliwiający załadowanie modułów ze sterownikami i wystartowanie reszty systemu z głównej partycji), aby załadować sterowniki systemu ext3 przed zamontowaniem głównej partycji. Bez tego system też wystartuje, ale główna partycja będzie zamontowana jako ext2 — żaden inny system plików z żurnalem nie jest tak dobroduszny.

Podczas kompilacji nowego jądra systemu należy upewnić się, że włączona została obsługa systemu plików ext3.

Tworzenie RAM-dysku startowego

Przed restartem systemu należy uruchomić program *mkinitrd*:

```
# mkinitrd initrd-2.4.18-3.img 2.4.18-3
```

Pierwszym argumentem jest plik z obrazem dysku startowego, który zostanie umieszczony w katalogu */boot*. Może on mieć dowolną nazwę, pokazana powyżej jest zgodna z przyjętymi konwencjami. Drugi argument przekazuje wersję jądra, dla którego jest dany plik. Nie musi to być wersja działająca w danej chwili; powinna jednak być to wersja, z której uruchamiany będzie system. Pozostałe możliwe opcje wyjaśnione są na odpowiedniej stronie podręcznika systemowego.

Następnie należy dodać odpowiednią linijkę do pliku */boot/grub/grub.conf* lub */etc/lilo.conf*, zależnie od używanego programu ładującego. Do pliku konfiguracyjnego GRUB należy poniżej linijki określającej plik jądra (zaczynającej się wyrazem `kernel`) wpisać linijkę:

```
initrd /boot/initrd-2.4.18-3.img
```

Z kolei do pliku konfiguracyjnego LILO należy w części dotyczącej interesującej nas wersji jądra dopisać:

```
initrd=/boot/initrd-2.4.18-3.img
```

a następnie przeładować program startowy (tylko w przypadku LILO):

```
# lilo
```

Dodatkową korzyścią jest fakt, że dowolny inny system operacyjny, jak BeOS czy Windows (po zainstalowaniu odpowiednich sterowników), może korzystać z partycji ext3, ponieważ dla tych systemów wyglądają one identycznie jak partycje ext2.

Przyjrzyjmy się jeszcze innym argumentom Red Hata przemawiającym za użyciem systemu ext3.

Dostępność

Podobnie jak w przypadku innych systemów plików z żurnalem, nie ma konieczności każdorazowego uruchamiania polecenia `fsck`; na domowym, 20-gigabajtowym dysku może to być nieco irytujące, ale można sobie wyobrazić kilkugodzinne działanie programu *fsck* na macierzy RAID o pojemności dajmy na to terabajta. Brak konieczności uruchamiania `fsck` jest wspólną cechą systemów plików z żurnalem. Czas potrzebny na doprowadzenie systemu plików do stanu używalności po niewłaściwym zamknięciu systemu nie jest zależny od rozmiaru partycji, lecz wyłącznie od ilości danych w żurnalu. System ext3 udostępnia kilka opcji umożliwiających między innymi określenie ilości przechowywanej w żurnalu informacji. Według Red Hat, Inc. zwykle na odczytanie żurnala i wprowadzenie zmian do właściwego systemu plików wystarcza około sekundy czasu.

Spójność danych a prędkość

Prędkość działania systemu plików nie idzie w parze z jego niezawodnością. Można wystawić dane na większe ryzyko uszkodzenia w razie niewłaściwego zamknięcia systemu, zyskując szybsze działanie systemu plików, lub poświęcić prędkość, zapewniając zgodność danych na dysku ze stanem systemu operacyjnego.

Dostępne są trzy tryby pracy systemu ext3:

- *writeback* — pozwala starym danym pozostać w systemie plików, osiągając tym samym maksymalną prędkość; nie zapisuje nic bezpośrednio na dysku, pozostawiając opróżnianie buforów kodowi jądra uaktywniającego się co 30 sekund,
- *ordered* — dane pozostają spójne, jednak kosztem częściowej utraty prędkości (domyślny tryb pracy w dystrybucji Red Hat),
- *journal* — wymaga większej ilości przestrzeni dyskowej, przechowując więcej danych w żurnalu; wydajność jest niższa, ponieważ dane zapisywane są dwukrotnie, ale prędkość może wzrosnąć przy synchronicznych zapisach charakterystycznych dla baz danych.

Dla większości użytkowników domyślny wybór jest dobrym kompromisem. Red Hat obsługuje start systemu z głównej partycji sformatowanej jako ext3 z odpowiednimi sterownikami ładowanymi ze startowego RAM-dysku.

Tryb pracy wybiera się, ustawiając odpowiednią opcję montowania w pliku */etc/fstab*.

System plików ReiserFS

Innym popularnym systemem plików z żurnalem jest pisany od zera system pomysłu Hansa Reisera, ReiserFS. Jest używany głównie w dystrybucjach SuSE i Mandrake, obsługujących start systemu z partycji ReiserFS. System ten udostępnia podobną funkcjonalność do ext3, ale nie ma prostego sposobu na konwersję partycji ext2 do formatu ReiserFS (należy utworzyć kopię danych, przeformatować partycję, a następnie odtworzyć dane z kopii zapasowej). System ten nie sprawdza się przy zdalnym montowaniu w protokole NFS[4]. W ReiserFS nie jest możliwe odzyskiwanie skasowanych plików.

Red Hat Linux obsługuje ReiserFS, lecz bez możliwości uruchomienia systemu z partycji sformatowanej w systemie plików Reisera. Nie umożliwia także podczas instalacji sformatowania partycji jako ReiserFS. Obsługa jest obecna w systemie głównie z myślą o dostępie do istniejących już partycji ReiserFS.

[4] Co więcej, jest (nie bez powodu) uważany za bardzo niestabilny; szczególnie nie lubi wielu plików w katalogu i intensywnego wykorzystywania dowiązań symbolicznych. Wartą uwagi cechą, o której autor nie wspomina, jest bezblokowość — dane nie są dzielone na bloki, lecz trzymane w drzewiastej strukturze danych; w systemie ReiserFS plik zajmuje dokładnie tyle miejsca na dysku, ile wynosi jego rozmiar. — *przyp. tłum.*

JFS i XFS

Zarówno SGI z systemem XFS, jak i IBM ze swoim JFS, udostępniają swoje systemy plików Linuksowi. Ponieważ przeznaczone są one głównie do profesjonalnych zastosowań, najprawdopodobniej są udostępniane, aby ułatwić użytkownikom systemów IRIX i AIX przejście na Linuksa bez konieczności reformatowania ogromnych systemów plików.

Żaden z tych systemów nie jest obsługiwany w Red Hat Linuksie 7.3, ale żądny przygód administrator może nałożyć odpowiednią łatę na jądro systemu i zdobyć narzędzia potrzebne do obsługi tych systemów plików. SGI udostępniało płytę CD umożliwiającą instalację dawniejszych wersji Red Hata z obsługą XFS; prawdopodobnie powstanie także płyta dla Red Hata 7.3.

Systemy plików DOS

Stopień obsługi DOS-owych systemów plików w Linuksie jest często zaskakujący dla nowych użytkowników, ale systemy te już w pierwszych latach Linuksa okazały się przydatną opcją. Microsoft był głównym producentem systemów operacyjnych dla intelowskich komputerów PC, a Linux zawsze skłaniał się w stronę współdziałania.

vfat, FAT12, FAT16 i FAT32

Systemy operacyjne DOS i Windows używają systemu plików FAT (*File Allocation Table*). Liczba po skrócie FAT określa rozmiar przestrzeni adresowej wskaźników; im większa przestrzeń, tym większy ciągły fragment dysku można zaadresować i obsłużyć. Wczesne wersje systemu FAT były tworzone z myślą o dyskietkach i nie mogły obsłużyć dużych urządzeń bez użycia niewydajnych, dużych bloków danych. System taki, FAT12, jest wciąż używany na dyskietkach (Linux bez większych problemów korzysta z dyskietek sformatowanych jako FAT12, chociaż systemy ext2 i minix też mają zastosowanie, gdy liczy się ilość miejsca dla danych pozostała po sformatowaniu dyskietki). Starsze wersje systemu FAT nie obsługują plików o nazwach dłuższych niż jedenaście znaków. Najnowszą wersją jest FAT32, zgodny wstecz z wcześniejszymi wersjami; starsze wersje nie są kompatybilne do przodu.

Jądro systemu załączone do dystrybucji Red Hat 7.3 może obsłużyć wszystkie wersje partycji FAT (włącznie z dyskietkami) przy użyciu modułu jądra `vfat`.

umsdos

Przede wszystkim system umsdos istnieje z tego samego powodu, z którego Linux z początku używał systemu plików miniksa. Był powszechnie dostępny i... działał. Używając systemu plików umsdos, można zainstalować Linuksa na partycji używanej przez system DOS lub Windows bez konieczności partycjonowania dysku. Używanie

dwóch systemów operacyjnych na jednym komputerze było codziennością wczesnych użytkowników i pionierów Linuksa. Ponadto dopiero niedawno stały się dostępne bezpieczne i niezawodne programy zmieniające rozmiar partycji bez utraty danych. Korzystanie z systemu plików umsdos było przydatne, wygodne, a do tego sprytne. Aby Linux mógł korzystać w środowisku wieloużytkownikowym z plików znajdujących się na partycji FAT, należało gdzieś przechować informacje i-węzłów związane z prawami dostępu. Niestety systemy plików z rodziny FAT nie umożliwiały przechowania takich informacji. Nigdy nie było takiej potrzeby — systemy te były zaprojektowane z myślą o jednoużytkownikowym systemie operacyjnym.

Modyfikacja systemu FAT umożliwiająca wykorzystanie go w wieloużytkownikowym środowisku Linuksa została nazwana umsdos. W każdym katalogu używanym przez Linuksa znajduje się plik *--linux-.---*, zawierający potrzebne Linuksowi informacje. Są one synchronizowane z bieżącym stanem plików za pomocą polecenia `/sbin/umssync`. Inicjalizacja istniejącego katalogu systemu DOS nazywana jest promocją katalogu; można ją przeprowadzić ręcznie lub automatycznie podczas montowania systemu plików.

Jako że obsługa systemu umsdos staje się w miarę pojawiania się kolejnych nowoczesnych alternatyw coraz mniej pożądana i przydatna, Red Hat Linux nie zawiera polecenia `umssync`; narzędzia obsługujące system umsdos można w razie potrzeby zdobyć niezależnie od dystrybucji, skompilować i zainstalować je samodzielnie.

Aby zamontować istniejącą partycję FAT jako partycję DOS, użyj typu vfat (może okazać się konieczne załadowanie modułu jądra `vfat.o`), aby widzieć długie nazwy plików, jeżeli montowany system je obsługuje. Jeżeli trzeba użyć bardziej restrykcyjnych uprawnień dla partycji DOS niż domyślne, można ustalić je, przekazując odpowiednie opcje do polecenia `mount`; jeżeli musisz ustawić uprawnienia oddzielnie dla poszczególnych plików, zamontuj dysk w systemie `umsdos`, dokonaj promocji katalogu i ustaw konieczne uprawnienia.

Systemy plików CD-ROM

Dla przeciętnego użytkownika zamontowana płyta CD-ROM wygląda jak część własnego systemu plików Linuksa. Faktycznie nie jest ona własnym systemem plików, ale dzięki funkcjonalności VFS wygląda, jakby była. Standardy systemów plików używanych na płytach CD-ROM wciąż ewoluują, obejmując coraz to nowe technologie.

iso9660

System plików zwykle używany na płytach CD-ROM jest znany pod nazwą standardu definiującego ten format, iso9660. Każdy system operacyjny tłumaczy system iso9660 na własny system plików (z pewnymi ograniczeniami). Utworzono kilka rozszerzeń standardu w celu wyjścia naprzeciw kilku powszechnym potrzebom. Rozszerzenie RockRidge udostępnia długie nazwy plików, uniksowe prawa dostępu i dowiązania symboliczne. Rozszerzenie Joliet udostępnia długie nazwy plików i obsługę znaków Unicode, przydatną przy językach innych niż angielski. Kompakty El Torito zawierają obraz startowy, umożliwiając przy wykorzystaniu odpowiedniego BIOS-u start systemu operacyjnego z płyty CD.

UDF

Ten system plików używany jest na płytach DVD; nazwa jest skrótem określenia *Universal Disk Format* — uniwersalny format dysku. Ma on wbudowanych wiele funkcji, do obsługi których system plików iso9660 potrzebowałby wprowadzenia rozszerzeń.

Tworzenie systemów plików

Gdy dysk twardy został już podzielony na partycje, należy utworzyć na nim system plików. Pierwsza z tych czynności jest określana w świecie systemu DOS jako formatowanie niskopoziomowe, a druga po prostu jako formatowanie. W świecie uniksowym jest to określane jako tworzenie systemu plików.

Niesformatowany dysk (dyskietka, dysk twardy lub napęd wymienny) z reguły dostarczany jest z gotowym niskopoziomowym formatem (z reguły tworzonym narzędziem, takim jak *fdisk* lub *superformat*). Może on posiadać sektor startowy, a nawet informacje o partycjach, ale z reguły nie zawiera struktury umożliwiającej przechowywanie plików. Jeżeli jest to coś więcej niż dyskietka, najpierw należy dostroić tablicę partycji za pomocą programu *fdisk* lub wybranego odpowiednika.

Dyski Zip z reguły posiadają jedną partycję o numerze 4; ma to jakieś tajemnicze znaczenie dla użytkowników komputerów firmy Apple, lecz dla użytkowników Linuksa nie ma żadnego. Większość informacji o dyskach Zip stwierdza, że montowana jest partycja numer 4 i tyle. Jeżeli kogoś to irytuje, może to łatwo zmienić.

Aby utworzyć strukturę właściwą dla danego systemu plików, musimy dokonać formatu wysokiego poziomu. W systemach DOS-owych służy do tego polecenie `format`; w Linuksie — polecenie `mke2fs` do utworzenia systemu plików ext2 lub ext3, `mkreiserfs` — dla systemu ReiserFS i `mkdosfs` — dla systemu FAT.

W systemie mogą być dostępne także inne polecenia, których używamy w razie potrzeby. Oto one:

- `mkdosfs`,
- `mke2fs`,
- `mkfs`,
- `mkfs.bfs`,
- `mkfs.ext2`,
- `mkfs.ext3`,
- `mkfs.minix`,
- `mkfs.msdos`,
- `mkfs.reiserfs`,
- `mkfs.vfat`,
- `mkreiserfs`.

mke2fs

Pełny przegląd wszystkich opcji polecenia `mke2fs` można znaleźć na stronie podręcznika systemowego polecenia `mke2fs`. Oto najbardziej przydatne argumenty, podane na tej stronie.

- Aby sprawdzić dysk w poszukiwaniu sektorów uszkodzonych podczas tworzenia systemu plików, użyj opcji `-c`.
- Użyj opcji `-N`, aby zmienić domyślną ilość tworzonych i-węzłów (z reguły służy to zwiększeniu dostępnego miejsca na dysku); domyślna wartość z reguły stanowi dobry wybór.
- Domyślnie system rezerwuje 5% bloków na wyłączny użytek superużytkownika do wykorzystania podczas sprawdzania systemu plików; wartość tę można zmniejszyć przy użyciu opcji `-m`, zwalniając trochę miejsca na dysku.
- Opcji `-L` można użyć w celu ustalenia etykiety systemu plików, przydatnej w razie konieczności przypomnienia sposobu wykorzystania systemu plików, gdy pracuje się z wieloma systemami plików, lub zapewnienia nieco większej elastyczności ustawień w pliku */etc/fstab*.
- Jako ostatniej próby odzyskania uszkodzonego systemu plików można użyć opcji `-s`, aby zapisać tylko superblok i deskryptory, a nie zmieniać zawartości i-węzłów; po zakończeniu działania użyj polecenia `fsck`.

Jak widać, kilka opcji umożliwia udostępnienie większej ilości miejsca dla zwykłych użytkowników kosztem superużytkownika, który mógłby wykorzystać to miejsce podczas sprawdzania systemu plików do odzyskania plików uszkodzonych. Chociaż marnotrawstwo nie jest pozytywną cechą, domyślne ustawienia są zadowalające dla większości użytkowników, a dyski twarde stają się coraz tańsze.

mkfs.ext3

Polecenie `mkfs.ext3` jest twardym dowiązaniem do polecenia `mke2fs`; wywołanie polecenia pod tą nazwą dodaje żurnal do tworzonego systemu plików. Można też użyć opcji `-j` lub `-J` polecenia `mke2fs` albo dodać żurnal do systemu ext2 za pomocą polecenia `tune2fs`:

```
# tune2fs /dev/hdxn -j
```

Dokładny opis tych opcji, wzięty ze strony podręcznika systemowego, podajemy niżej.

- `-j` — dodaje do systemu plików żurnal. Jeżeli nie podano opcji `-J`, do utworzenia odpowiednich rozmiarów żurnalu w systemie plików zostaną użyte wartości domyślne (wyliczone na podstawie rozmiaru systemu plików). Aby korzystać z żurnalu, należy uaktywnić w jądrze systemu obsługę systemu plików ext3.
- `-J` *opcje_żurnalu* — podaje parametry tworzonego żurnalu. Opcje podaje się rozdzielone przecinkiem, z wartością podaną po znaku =; obsługiwane są następujące opcje:

- `size=`*`rozmiar_żurnalu`* — tworzy w systemie plików żurnal o podanym (w megabajtach) rozmiarze, rozmiar musi być równy co najmniej 1024 bloki systemu plików (czyli 1 MB dla 1-kilobajtowych bloków, 4 MB dla bloków 4-kilobajtowych itd.), ale nie większy nić 102.400 bloki; w systemie plików musi być wystarczająca ilość wolnego miejsca,
- `device=`*`żurnal_zewnętrzny`* — ta opcja uaktywni w systemie pliku zewnętrzny żurnal; musi on być utworzony poleceniem `mke2fs -O journal` *`żurnal_zewnętrzny`*; żurnal i urządzenie zawierające dane nie muszą znajdować się na tym samym urządzeniu.

Można użyć tylko jednej opcji naraz — wykluczają się one wzajemnie.

Tryb korzystania z żurnala przez system ext3 można wybrać, dodając odpowiedni wpis do pliku */etc/fstab*. Oto fragment strony podręcznika systemowego polecenia `mount`, opisujący opcje montowania systemu ext3.

System plików ext3 jest wersją systemu ext2 poszerzoną o obsługę żurnalu. Obsługuje wszystkie opcje obsługiwane przez ext2, a ponadto:

- `journal=update` — uaktualnia format żurnalu ext3,
- `journal=`*`iwęzeł`* — jeżeli w systemie plików istnieje już żurnal, opcja jest ignorowana; jeżeli nie, zawartość i-węzła o podanym numerze zostaje przeznaczona na żurnal; sterownik utworzy nowy żurnal, nadpisując dotychczasową zawartość pliku,
- `noload` — nie ładuje żurnala ext3 podczas montowania,
- `data=journal` / `data=ordered` / `data=writeback` — określa tryb korzystania z żurnalu dla danych pliku (metadane są zawsze zapisywane w żurnalu):
  - `journal` — dane są zapisywane w żurnalu przed zapisaniem ich we właściwym systemie plików,
  - `ordered` — tryb domyślny, dane są kierowane do głównego systemu plików przed zapisaniem metadanych w żurnalu,
  - `writeback` — kolejność danych nie jest zachowana, dane mogą być zapisane do właściwego systemu plików po zapisaniu metadanych w żurnalu, ponoć daje to największą wydajność, jednak po awarii systemu w plikach mogą znaleźć się stare dane.

mkreiserfs

Dokładny opis wszystkich opcji, argumentów i składni polecenia do tworzenia systemu plików Reisera można znaleźć na stronie podręcznika systemowego polecenia `mkreiserfs`. Składnia i argumenty przedstawiają się następująco:

```
mkreiserfs [ -h r5 | tea | rupasov ] [ -v 1 | 2 ] [ -q ] urządzenie [ rozmiar-w-blokach ]
```

- `-h r5 | tea | rupasov` — wybiera funkcję skrótu używaną do sortowania plików w katalogach. Należy wybrać jedną z podanych. Domyślną jest `r5`.
- `-v 1 | 2` — wybiera format tworzonego systemu plików.
- `-q` — zmniejsza ilość wypisywanych informacji (przydatne, gdy jesteś zalogowany na wolnym łączu).

Podczas formatowania partycji ReiserFS, gdy jesteś zalogowany za pomocą wolnego łącza, przydaje się opcja `-q`, zmniejszająca szczegółowość informacji wyświetlanej w pasku postępu. Dokładniejsze informacje można znaleźć na stronie podręcznika systemowego; w większości przypadków wartości domyślne są wystarczające.

mkdosfs

Można łatwo utworzyć system plików FAT, nie posiadając żadnego oprogramowania Microsoftu. Dokładne omówienie wszystkich opcji linii poleceń i składni do tworzenia DOS-owego systemu plików znajduje się na stronie podręcznika systemowego polecenia `mkdosfs`.

Oto niektóre z przydatniejszych argumentów:

- opcja `-c` włącza sprawdzanie dysku w poszukiwaniu uszkodzonych sektorów,
- gdy potrzebny jest plik z obrazem dysku FAT do celów testowych, można użyć opcji `-C`, podając nazwę pliku do utworzenia i wymaganą ilość bloków; byłoby to miłą opcją dla innych poleceń z rodziny `mkfs`,
- `mkdosfs` umie automatycznie wybrać odpowiednią wersję systemu FAT podczas tworzenia systemu plików, ale podanie opcji `-F`, a następnie `12`, `16` lub `32` pozwala sforsować własny wybór,
- można podać etykietę dysku, o długości do 11 znaków, podając ją po opcji `-n`,
- opcja `-v` włącza szczegółowe informacje wyjściowe, udostępniając dodatkowe informacje.

Do sprawdzenia spójności systemu plików FAT można użyć polecenia `dosfsck`.

Montowanie systemów plików

Systemy plików w systemach uniksowych są na tyle elastyczne, że nie muszą być fizycznie obecne w danym komputerze, można zamontować zdalny dysk za pośrednictwem sieci. Linuksowy system VFS sprawia, że wszystkie systemy plików widoczne są jako lokalna część głównego drzewa katalogów. Administrator systemu musi zdecydować, jakie systemy plików udostępnić i w którym miejscu przyłączyć je — zamontować — do głównego drzewa katalogów.

Dlaczego systemy plików trzeba montować?

W Linuksie (jak i w innych systemach uniksowych) wszystkie systemy plików — lokalne, zdalne, przechowywane na dysku bądź w pamięci — są zamontowane we wspólnym drzewie katalogów, rozpoczynającym się od katalogu głównego (nazywanego *root directory*, chociaż nie ma on nic wspólnego z użytkownikiem *root*). Katalog ten określa się pojedynczym ukośnikiem, /. Po zamontowaniu fizyczna lokalizacja systemu plików nie ma znaczenia.

Gdzie montuje się systemy plików?

Każdy system plików może być zamontowany w dowolnym miejscu drzewa katalogów, ale niektóre z miejsc bywają bardziej przydatne niż inne. Nawet jeżeli zamontowane są różne systemy plików (FAT, ext2, HPFS, ntfs itd.), podsystem VFS jądra Linuksa sprawia, że są one widoczne w drzewie katalogów jako własne pliki systemu.

Polecenie mount

Systemy plików montuje się za pomocą polecenia `mount`, a wymontowuje przy użyciu polecenia `umount`. Powód, dla którego drugie polecenie nie nazywa się `unmount`, ginie w mrokach historii systemów uniksowych; nazywa się ono tak, jak się nazywa, i trzeba się do tego przyzwyczaić, czy się to wydaje sensowne, czy nie. Polecenie `mount` korzysta z pliku */etc/fstab* — tablicy systemów plików, zawierającej informacje o systemach plików stale obecnych w systemie, punktach ich zamontowania oraz opcjach montowania. Ogólna składnia polecenia `mount` jest następująca:

```
# mount -t typ urządzenie punkt_zamontowania
```

Oto wyjaśnienie poszczególnych elementów składni polecenia `mount`.

- `typ` — zawsze poprzedzony opcją `-t` oraz spacją typ montowanego systemu plików. Można podać między innymi `ext2`, `vfat`, `iso9660`, `umsdos`, `msdos`, `hpfs`, `hfs`, `ntfs`. Argument ten jest często zbędny, ponieważ `mount` jest w stanie automatycznie wykryć rodzaj systemu plików.
- `urządzenie` — urządzenie zawierające montowany system plików, zwykle będzie to coś w stylu `/dev/hd`*xn*, `/dev/sd`*xn* lub `/dev/fd`*x*.
- `punkt_zamontowania` — miejsce w hierarchii katalogów, w którym zostanie zamontowany system plików. Dobrym miejscem do tymczasowego zamontowania systemu plików jest podkatalog katalogu */mnt*. Można utworzyć specjalny katalog dla zamontowania systemu plików w najbardziej niespodziewanym miejscu. Co ciekawsze, system plików można zamontować „nad" istniejącym już katalogiem — na przykład jeżeli w katalogu */foo* mamy plik *bar*, a następnie w */foo* zamontujemy system plików zawierający plik *snafu*, będziemy mogli zobaczyć plik *snafu*, a plik *bar* będzie niewidoczny. Możliwość dostępu do obydwu plików nazywana jest przezroczystym montowaniem systemu plików i nie jest obsługiwana przez Linuksa.

Jedynym ograniczeniem „montowania czegokolwiek gdziekolwiek" jest fakt, że niezbędne do pracy systemu pliki z katalogów */bin*, */sbin*, */lib*, */etc*, */dev*, */proc* i */tmp* muszą być obecne podczas startu systemu, więc powinny znajdować się na tym samym dysku, montowanym jako główny system plików (katalog */*). Jeżeli pliki te są nieobecne podczas startu systemu, Linux nie uruchomi się.

umount

Aby wymontować system plików, należy użyć polecenia `umount`:

```
# umount punkt_zamontowania
```

Można też przekazać nazwę zamontowanego urządzenia:

```
# umount /dev/urządzenie
```

Można wymontować wszystkie systemy plików, które nie są niezbędne do pracy systemu (ani nie są w danej chwili używane) poleceniem `umount -a`, ale nie jest to dobrym pomysłem na wieloużytkownikowym systemie sieciowym, ponieważ wtedy użytkownicy stracą dostęp do części lub wszystkich swoich plików. Więc, jak powiedziałby każdy dobry administrator, nie należy tego robić.

Do polecenia `mount` można przekazywać opcje montowania systemu plików, podając je (jako listę rozdzieloną przecinkami) po opcji `-o`, są one jednak używane głównie w pliku */etc/fstab*. Spis dostępnych opcji znajduje się na stronie podręcznika systemowego polecenia `mount`; najczęściej używane wymienione są w dalszej części tego rozdziału.

Automatyczne montowanie — plik fstab

Plik systemowy */etc/fstab* zawiera informacje na temat systemów plików, punktów i opcji montowania, aby umożliwić automatyczne montowanie systemów plików podczas startu systemu, a także przyspieszyć montowanie ręczne, eliminując konieczność wpisywania powtarzających się zestawów opcji. Dla różnych systemów plików można użyć różnych opcji; szczególnie częstym ich zastosowaniem jest ustawienie praw dostępu do plików w systemach FAT, jako że sam system plików nie udostępnia takiej funkcjonalności.

Plik */etc/fstab* może zapisywać jedynie superużytkownik. Korzystają z niego polecenia, takie jak `fsck`, `mount` i `umount`. Opis każdego z systemów plików zajmuje jedną linijkę; poszczególne pola wpisu rozdzielone są spacjami.

Pełny opis składni pliku znajduje się na stronie podręcznika systemowego *fstab*. Poniżej przedstawiony jest opis skrócony.

W każdej linijce pierwsze pole określa urządzenie lub adres zdalnego systemu plików do zamontowania.

W drugim polu znajduje się punkt zamontowania systemu plików w lokalnym drzewie katalogów.

Trzecie pole określa typ systemu plików.

Czwarte pole jest listą opcji, rozdzielonych przecinkami. Często zawierają one opcje `noauto` (system plików nie jest montowany automatycznie przy starcie systemu) i `user` (wskazujący, że dany system plików może być montowany przez użytkownika; z reguły ta opcja jest włączona dla napędów CD-ROM i dyskietek). Oto najważniejsze z pozostałych opcji:

- `exec` — zezwala na wykonywanie programów,
- `noauto` — system plików nie będzie montowany automatycznie (nie dotyczy go opcja `-a`),
- `noexec` — blokuje wykonywanie programów; może to być przydatne na serwerze z zamontowaną partycją zawierającą pliki wykonywalne dla innej architektury,
- `nosuid` — blokuje działanie bitów SUID i SGID (daje to poczucie bezpieczeństwa, ale jest ono złudne, jeżeli w systemie zainstalowane są programy, takie jak chociażby *suidperl*),
- `ro` — montuje system tylko do odczytu,
- `rw` — montuje system do odczytu i zapisu,
- `sync` — wszystkie operacje wejścia-wyjścia mają być wykonywane synchronicznie (bez buforowania),
- `user` — umożliwia użytkownikom montowanie systemu plików; ta opcja włącza automatycznie opcje `noexec`, `nosuid` i `nodev` (chyba że zostaną wyłączone przez późniejsze podanie opcji przeciwnych, np. `user,exec,dev,suid`).

Opcjami systemu plików FAT są m. in.:

- `umask=`*`wartość`* — ustawia maskę dostępu (czyli bity uprawnień, które mają być wyłączone); domyślną wartością jest maska dostępu bieżącego procesu; maskę podaje się jako liczbę ósemkową,
- `conv=b[inary]` / `conv=t[ext]` / `conv=a[uto]` — sterownik systemu FAT może na bieżąco przekształcać pliki z DOS-owym zapisem nowej linii (CRLF) na pliki z zapisem uniksowym (NL) i na odwrót; dostępne są następujące tryby konwersji:
  - `binary` — konwersja nie jest dokonywana (ten tryb jest domyślny),
  - `text` — konwersja przeprowadzana jest dla wszystkich plików,
  - `auto` — konwersja przeprowadzana jest dla plików o znanym rozszerzeniu tekstowym; lista znanych rozszerzeń znajduje się w pliku */usr/src/linux/fs/fat/misc.c* i w jądrze dystrybucji Red Hat Linux wygląda ona tak:

```
static char ascii_extensions[] =
  "TXT" "ME " "HTM" "1ST" "LOG" "   "   /* text files */
  "C  " "H  " "CPP" "LIS" "PAS" "FOR"  /* programming languages */
  "F  " "MAK" "INC" "BAS"              /* programming languages */
  "BAT" "SH "                          /* program code :) */
  "INI"                                /* config files */
  "PBM" "PGM" "DXF"                    /* graphics */
  "TEX";                               /* TeX */
```

Interesującą opcją systemu plików iso9660 jest `unhide`, pokazująca normalnie niewidoczne pliki.

Pozostałe pola

Piąte pole używane jest przez program *dump* (do wykonywania kopii zapasowych) w celu określenia, czy powinna zostać utworzona kopia zapasowa tego systemu plików; `1` oznacza „tak”, `0` — „nie”.

Szóste pole jest używane przez program *fsck* w celu określenia częstotliwości i kolejności sprawdzania systemu plików. `0` oznacza, że ten system nie będzie automatycznie sprawdzany (przydatne dla systemów FAT); `1` — że system będzie sprawdzany w zadanym czasie. Zaleca się użyć `2` dla niegłównych systemów plików, aby *fsck* sprawdzał je rzadziej.

Przykładowy plik fstab

Oto zawartość prostego pliku *fstab* systemu z partycją główną w systemie plików ext3 na macierzy RAID0 i uruchamianego wymiennie z MS Windows:

```
LABEL=/                 /                       ext3    defaults        1 1
none                    /dev/pts                devpts  gid=5,mode=620  0 0
none                    /proc                   proc    defaults        0 0
none                    /dev/shm                tmpfs   defaults        0 0
/dev/hda5               swap                    swap    defaults        0 0
/dev/fd0                /mnt/floppy             auto    noauto,owner,kudzu 0 0
/dev/cdrom              /mnt/cdrom              iso9660 noauto,owner,kudzu,ro 0 0
/dev/hda1               /mnt/win_c              vfat    defaults,quiet,exec 0 0
```

Dwie linijki oznaczone są opcją `kudzu`. Jest to efekt działania polecenia `updfstab`, który wykrywa i dopisuje do pliku *fstab* urządzenia wymienne, jak stacje dyskietek, napędy CD-ROM, Zip, Jaz, LS-120 czy niektóre kamery cyfrowe. Opcja `quiet` przy partycji Windows powstrzymuje ostrzeżenia; zaleca się włączenie tej opcji, gdy używa się programu Wine.

Edycja pliku fstab

Do edycji pliku *fstab* wystarczy dowolny edytor plików tekstowych, uprawnienia superużytkownika i znajomość składni pliku i znaczenia poszczególnych opcji.

Macierze RAID

Programowe macierze RAID są w systemie Red Hat Linux 7.3 zaskakująco łatwe w konfiguracji zarówno w czasie instalacji, jak i później. Efektywne wykorzystanie macierzy RAID wymaga zrozumienia ich działania i wcześniejszego przemyślenia konfiguracji.

Sprzętowe macierze RAID, korzystające z zewnętrznego lub wbudowanego w płytę główną kontrolera, stają się coraz tańsze w miarę spadku cen i wzrostu prędkości dysków IDE. Z coraz lepszą obsługą i nowymi sterownikami kontrolerów RAID macierze sprzętowe stają się coraz powszechniej wykorzystywaną opcją także dla użytkowników chcących wykorzystać je ze względu na wydajność.

Czym jest RAID?

Macierz RAID nie jest metodą ochrony przed uszkodzeniem danych (zapisze ona na dysk dane uszkodzone w taki sam sposób, jak właściwe). Z drugiej strony, niektóre poziomy RAID mogą uchronić przed utratą danych, zwiększyć wydajność albo połączyć kilka dysków fizycznych w jeden dysk logiczny. Przeciętny indywidualny użytkownik nie potrzebuje macierzy RAID, bowiem system domowy rzadko kiedy jest systemem krytycznym (chociaż odczucia po utracie plików do pracy lub szkoły mogą być nieco inne). Większa wydajność macierzy RAID nie jest aż tak istotna (większość sprzętu jest wystarczająco szybka do zastosowań domowych), a cena dużych dysków spadła tak nisko, że nie ma potrzeby łączenia mniejszych dysków w jeden większy (chyba że kolekcja filmów lub plików MP3 naprawdę wymknęła się spod kontroli). Najprostszy, liniowy poziom RAID wydaje się być lepiej obsługiwany przez system LVM.

Oto definicja poszczególnych poziomów RAID.

- Tryb liniowy — kilka dysków jest łączonych w jedno wirtualne urządzenie. Są one zwyczajnie łączone jeden po drugim, tak że najpierw zapełniany jest pierwszy dysk, potem następny i tak dalej.
- RAID 0 — tryb przeplotu. Dane są zapisywane równolegle na kilku urządzeniach o pojemności w przybliżeniu równej pojemności. Daje to pewne zyski w wydajności, ale nie chroni danych.
- RAID 1 — tryb lustrzany. Dane są zapisywane na kilku dyskach. Działa to wolniej niż w przypadku pojedynczego dysku, ale oferuje pewną ochronę danych. Jeżeli dane ulegną uszkodzeniu, zostają odtworzone z pozostałej kopii.
- RAID 4 — działa podobnie jak RAID 0, ale potrzebuje co najmniej trzech dysków i na jednym z nich trzyma dane parzystości używane do ochrony danych. Każda operacja zapisu zapisuje także dysk parzystości, więc powinien to być najszybszy dysk w zestawie.
- RAID 5 — działa podobnie jak RAID 4, ale dane parzystości rozproszone są na różnych dyskach, co poprawia wydajność i umożliwia odzyskanie danych po jednoczesnej awarii kilku dysków.

- RAID 10 — macierz RAID-1 dwóch macierzy RAID-0.
- Partycja wymiany RAID — jest to dostępna od pewnego czasu funkcjonalność jądra. Wystarczy utworzyć na kilku dyskach partycje wymiany i wpisać je do plik */etc/fstab*, nadając im równy priorytet przez dopisanie opcji `pri=0`.

Użycie macierzy RAID wymaga wstępnego zaplanowania zakupu sprzętu, a także dokładnej lektury odpowiedniego dokumentu HOWTO w celu lepszego zrozumienia dokonywanego wyboru. Część programu instalacyjnego Red Hata Anaconda dotycząca partycjonowania dysku umożliwi łatwą konfigurację macierzy RAID podczas instalacji.

Przenoszenie systemu plików

Większość użytkowników indywidualnych rozpoczyna pracę z Linuksem od pojedynczej partycji zawierającej całe drzewo katalogów. Jest to wystarczające do zastosowań domowych, ale po pewnym czasie dysk może okazać się za mały. Dodanie nowego dysku i przeniesienie na niego części systemu plików nie jest trudne, ale jest problematyczne dla wielu nowych użytkowników Linuksa. W poniższym przykładzie zainstalujemy nowy dysk jako */dev/hdb* (napęd *slave* na pierwszej taśmie), utworzymy na nim jedną partycję, sformatujemy ją w systemie ext3 i przeniesiemy tam dane użytkowników z katalogu */home*. Następnie zamontujemy nowy dysk jako katalog */home*.

Najpierw należy fizycznie zainstalować dysk, upewniając się, że zworki *master*/*slave* są odpowiednio ustawione (nasz dysk powinien być ustawiony jako *slave*) zarówno na dodawanym, jak i na starym dysku (niektóre dyski wymagają innego ustawienia zwór, gdy są jedynym dyskiem na kablu, a innego, gdy są dyskiem *master* z dyskiem *slave* na tej samej taśmie). Nieprawidłowe uzworkowanie dysków jest błędem popełnianym nawet przez ludzi obeznanych ze sprzętem. Po podłączeniu dysk powinien zostać wykryty przez BIOS. Większość BIOS-ów ma możliwość autodetekcji, można też wprowadzić geometrię ręcznie (ilość cylindrów, głowic i sektorów powinna być podana na obudowie lub w dokumentacji dysku).

Większość nowoczesnych dużych dysków korzysta z adresowania LBA, aby obejść ograniczenia rozmiaru dysków nałożone przez BIOS. Jeżeli BIOS nie wykrywa dysku, należy sprawdzić po kolei: kabel zasilania, poprawność podłączenia taśmy danych (zwykle żyła oznaczona na czerwono powinna być przy kablu zasilania, ale nie zaszkodzi sprawdzić w obie strony) i zwory *master*/*slave*. Jeżeli wszystko jest w porządku, dysk może być uszkodzony albo dwa dyski (jeżeli pochodzą od różnych producentów) mogą źle współpracować. Aby to sprawdzić, można odłączyć główny dysk i wpiąć nowy, ze zworką ustawioną na *master*, na jego miejsce. Jeżeli wtedy dysk zostanie wykryty, można podejrzewać niezgodność napędów. Należy pamiętać, żeby przełączać kable przy wyłączonym komputerze — w przeciwnym wypadku można uszkodzić dysk.

Po podłączeniu dysku i wykryciu przez BIOS należy utworzyć tablicę partycji. Używając programu *fdisk* (lub innego wybranego programu), tworzymy jedną partycję, obejmującą cały dysk. Trzeba pamiętać o zapisaniu zmian w MBR podczas wychodzenia z programu.

Następnie należy sformatować partycję. Ponieważ tworzymy system plików ext3, używamy polecenia:

```
# mke2fs -c -j /dev/hdb1
```

Włączyliśmy tu szukanie uszkodzonych sektorów dysku podczas formatowania. Program zlokalizuje te sektory i nie będzie przechowywał w nich danych; gdyby nie zostały wykryte, ryzykowalibyśmy uszkodzenie danych. Szukanie uszkodzonych sektorów znacząco spowalnia tworzenie systemu plików, ale jest zwykle dobrym pomysłem.

Utwórzmy teraz tymczasowy punkt montowania dysku i zamontujmy go:

```
# mkdir /mnt/nowapartycja
# mount -t ext3 /dev/hdb1 /mnt/nowapartycja
```

Teraz należy skopiować zawartość katalogu */home* na nowo utworzoną partycję. Istotne jest zachowanie znaczników czasu oraz praw dostępu. Kopiujemy całe drzewo katalogów; z trzech metod, których można użyć do tego celu (wykorzystanie poleceń `tar`, `cpio` lub `cp`), wybieramy najbardziej nam odpowiadającą:

```
# cp -a /home/* /mnt/nowapartycja
```

Następnie trzeba dopisać do pliku */etc/fstab* linijkę opisującą sposób montowania nowego systemu plików:

```
/dev/hdb3    /home    ext3    defaults    1  1
```

Wybraliśmy pozostawienie domyślnych opcji montowania dla naszej partycji. Są one identyczne jak domyślne opcje montowania systemu plików ext2, plus domyślny tryb używania żurnalu `data=ordered`.

Po restarcie systemu nowa partycja zawierająca skopiowane pliki będzie widoczna jako katalog */home*. Zanim to zrobimy, wejdźmy do katalogu */home* i utwórzmy nowy plik:

```
# touch tojeststarykataloghome
```

Teraz możemy próbnie zamontować nową partycję:

```
# umount /mnt/nowapartycja
# mount /dev/hdb1 /home
```

Teraz, jeżeli sprawdzimy zawartość katalogu */home*

```
# ls -al /home
```

nie zobaczymy pliku *tojeststarykataloghome* utworzonego poleceniem `touch`. Stare pliki dalej są na swoim miejscu, zostały jedynie przesłonięte zamontowaną partycją. Kiedy już sprawdzimy, że wszystko działa, możemy odmontować nową partycję i skasować starą zawartość katalogu */home* (który możemy rozpoznać po obecności pliku *tojeststarykataloghome*). Możemy użyć podobnej techniki do tworzenia plików „zastępczych" lub ostrzegających, że system plików, który powinien być w tym miejscu, nie jest zamontowany.

LVM

Najwyższy czas zacząć przyzwyczajać się do czegoś nowego. W poprzednim przykładzie użyliśmy tradycyjnego przykładu kończącego się miejsca na dysku i dodawania nowego napędu. A gdyby można to było zrobić w przezroczysty sposób? Tak właśnie działa LVM (*Logical Volume Manager* — menedżer woluminów logicznych): przestrzeń dyskowa z różnych urządzeń może być połączona w pojedynczy wolumin logiczny.

Podobnie jak w przypadku każdej nowej technologii, mamy tu stromą krzywą uczenia, na co składa się w dużej mierze słownictwo. To, co dotychczas nazywaliśmy partycjami, określimy teraz jako woluminy fizyczne; możemy dodawać woluminy fizyczne do listy woluminów składających się na wolumin logiczny, na którym możemy utworzyć system plików.

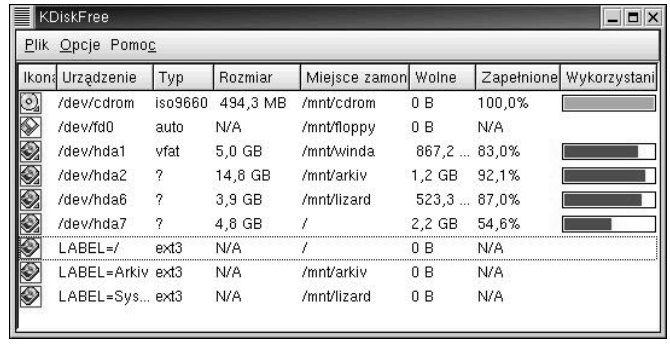
LVM może także tworzyć migawki woluminu logicznego, które można oddzielnie zamontować i wykonać kopię zapasową. Może się to wydawać bezużyteczne, ale na intensywnie używanych systemach pliki mogą się zmieniać w trakcie wykonywania kopii zapasowej, w związku z czym odtworzone później pliki mogą być błędne.

Szczegółowe informacje na temat konfiguracji systemu LVM znajdują się na stronie podręcznika systemowego *lvm*.

Narzędzia graficzne do montowania systemów plików

Program KDE KDiskFree (rysunek 10.1), uruchamiany z menu KDE lub z linii poleceń jako `kdf`, wyświetla wszystkie systemy plików wymienione w */etc/fstab*, pokazując informacje na ich temat i umożliwiając łatwe montowanie i wymontowywanie[5]. Inny program KDE, KwikDisk, umożliwia montowanie i wymontowywanie systemów plików przy użyciu apletu panela KDE (jest on apletem programu KDiskFree).

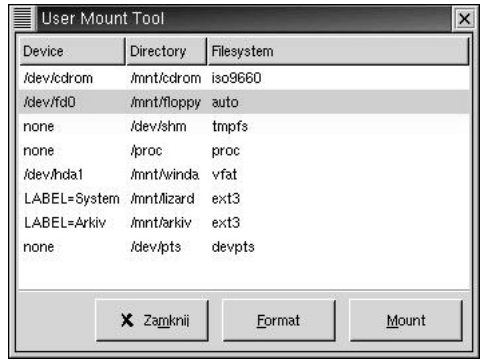
Rysunek 10.1.
Program KDiskFree



[5] Jak widać na rysunku, program nie radzi sobie szczególnie dobrze z dyskami opisanymi w `/etc/fstab` przez podanie etykiety zamiast pliku urządzenia — *przyp. tłum.*

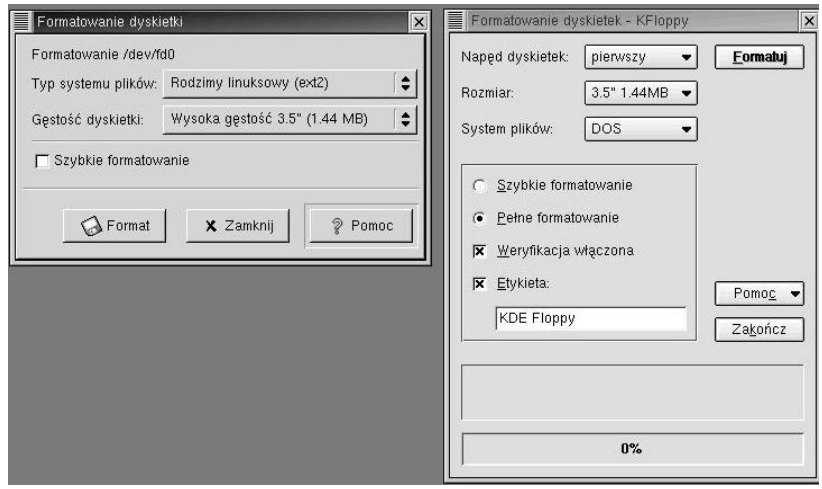
Narzędzie użytkownika montowania dysku, *usermount* (rysunek 10.2), znajduje się w menu KDE i GNOME w dziale *System*, pod nazwą *Disk Management*. Pozwala ono także formatować dyski.

Rysunek 10.2. *Narzędzie usermount*



Zarówno KDE, jak i GNOME zawierają programy do formatowania dyskietek. Program *gfloppy* znajduje się w menu GNOME *Narzędzia* pod nazwą *Formater dyskietek*, natomiast *kfloppy* można znaleźć w menu KDE *Użytki*. Oba programy pozwalają sformatować dyskietki 3,5" i 5,25" niskiej i wysokiej gęstości zarówno w systemie plików FAT, jak i ext2. Przedstawione są na rysunku 10.3.

Rysunek 10.3. *Programy do formatowania dyskietek, od lewej — dla GNOME i KDE*



Program e2label

Polecenie `e2label` może wyświetlić lub zmienić etykietę dysku ext2 lub ext3 (można to osiągnąć także opcją `-L` polecenia `tune2fs`). Aby na przykład zmienić etykietę partycji */dev/hda4* na „Dane", możemy powiedzieć:

```
# e2label /dev/hda4 Dane
```

Do czego może się przydać etykieta? Można użyć jej zamiast nazwy urządzenia w */etc/fstab* i, jeżeli stosuje się różne partycje, łatwiej jest utrzymać porządek, gdy mają one nazwy, a nie

numery. Można wtedy także zamieniać partycje miejscami bez zmieniania pliku *fstab*. Polecenie `e2label` jest łatwiejsze do zapamiętania niż opcja polecenia `tune2fs` (jakby jakiekolwiek polecenia uniksowe były łatwe do zapamiętania).

Przykłady

Naukowcy zajmujący się nauczaniem twierdzą, że różni ludzie uczą się na różne sposoby. Dla tych czytelników, którzy wolą przykład od wykładu, zaraz przedstawimy kilka.

Tworzenie testowego systemu plików

Ponieważ większość z nas nie posiada wolnego komputera lub dysku twardego do eksperymentów i ćwiczeń, utwórzmy więc „dysk" w pliku, zapisując do niego obraz systemu plików i montując go przy użyciu urządzeń-pętli (*loopback device*). W ten sposób nie ryzykujemy przypadkowego uszkodzenia właściwego systemu. W zasadzie można by też użyć dyskietek, ale ich mały rozmiar ogranicza możliwości ćwiczeń.

Krok pierwszy — tworzenie pustego pliku

Do utworzenia pliku użyjemy polecenia `dd`. Ustawimy rozmiar bloku na 1024 bajty (jeden kilobajt) i utworzymy plik o rozmiarze 10 MB (będzie konieczne posiadanie wystarczającej ilości wolnego miejsca na dysku). Potrzebujemy więc 10.240 kilobajtowej wielkości bloków.

Jeżeli obraz miałby mieć rozmiar dyskietki, należałoby wybrać 2880 bloków 512-bajtowych dla dyskietki 1,44 MB lub 5760 dla dyskietki 2,88 MB.

```
# dd if=/dev/zero of=/tmp/test.img bs=1024 count=10240
10240+0 rekordów wczytanych
10240+0 rekordów zapisanych
```

Jeżeli sprawdzimy typ pliku za pomocą polecenia `file`, zobaczymy następujący komunikat:

```
# file /tmp/test.img
/tmp/test.img: data
```

Krok drugi — tworzenie systemu plików

Mamy więc plik pełen zer. Należy teraz przekonać system, że jest to urządzenie blokowe, a nie plik z danymi, więc użyjemy polecenia `losetup`, przypisującego zwykłym plikom urządzenia-pętle:

```
# losetup /dev/loop0 /tmp/test.img
```

Następnie możemy utworzyć system plików:

```
# mke2fs /dev/loop0
mke2fs 1.27 (8-Mar-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
2560 inodes, 10240 blocks
512 blocks (5.00%) reserved for the super user
First data block=1
2 block groups
8192 blocks per group, 8192 fragments per group
1280 inodes per group
Superblock backups stored on blocks:
        8193

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 26 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Krok trzeci — montowanie testowego systemu plików

Gdy już utworzyliśmy system plików, możemy poeksperymentować z różnymi opcjami polecenia formatującego. Teraz przyda się punkt zamontowania dla naszego systemu plików:

```
# mkdir /mnt/image
```

Możemy teraz zamontować nasz plik:

```
# mount /dev/loop0 /mnt/image
```

Można zamontować plik w ten sposób, ponieważ do pliku jest już przypisane urządzenie-pętla. Przy późniejszym montowaniu należy zaznaczyć, że system powinien użyć urządzenie-pętli przez podanie opcji montowania `loop`:

```
# mount -o loop /tmp/test.img /mnt/image
```

Po zamontowaniu nowego systemu plików możemy do niego zajrzeć i zobaczyć, że utworzony został katalog *lost+found* i polecenie `df` zwraca wynik podobny do przedstawionego niżej:

```
# df -h /mnt/image
Filesystem            Size  Used Avail Use% Mounted on
/dev/loop0            9.7M   13k  9.1M   1% /mnt/image
```

Aby odmontować system plików, należy użyć polecenia:

```
# umount /mnt/image
```

Można też zrobić kopię zapasową pliku, na wypadek gdybyśmy popsuli oryginał:

```
# cp /tmp/test.img /tmp/test.img.bak
```

Po utworzeniu systemu plików można tworzyć w nim katalogi, kopiować do niego pliki, kasować je, próbować odzyskiwać i generalnie wprowadzać kontrolowany chaos, jednocześnie ucząc się i ćwicząc przydatne umiejętności. W razie zniszczenia systemu plików bez możliwości naprawy, należy go wymontować, skasować plik i utworzyć nowy.

Polecenie dumpe2fs

Sprawdźmy nasz system za pomocą polecenia `dumpe2fs` po uprzednim wymontowaniu:

```
# dumpe2fs /tmp/test.img
dumpe2fs 1.27 (8-Mar-2002)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          a1c778d8-fb16-4de5-b4df-51f4ed10bd3d
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      filetype sparse_super
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              2560
Block count:              10240
Reserved block count:     512
Free blocks:              9898
Free inodes:              2549
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         1280
Inode blocks per group:   160
Last mount time:          Sat Aug 10 00:45:14 2002
Last write time:          Sat Aug 10 00:53:07 2002
Mount count:              1
Maximum mount count:      26
Last checked:             Sat Aug 10 00:42:31 2002
Check interval:           15552000 (6 months)
Next check after:         Wed Feb  5 23:42:31 2003
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128


Group 0: (Blocks 1-8192)
  Primary Superblock at 1,  Group Descriptors at 2-2
  Block bitmap at 3 (+2), Inode bitmap at 4 (+3)
  Inode table at 5-164 (+4)
  8015 free blocks, 1269 free inodes, 2 directories
  Free blocks: 178-8192
  Free inodes: 12-1280
Group 1: (Blocks 8193-10239)
  Backup Superblock at 8193,  Group Descriptors at 8194-8194
  Block bitmap at 8195 (+2), Inode bitmap at 8196 (+3)
```

```
Inode table at 8197-8356 (+4)
1883 free blocks, 1280 free inodes, 0 directories
Free blocks: 8357-10239
Free inodes: 1281-2560
```

Wyświetla ono, jak widać, sporo informacji. Czytelnik może porównać je z opisami budowy systemu plików we wcześniejszej części rozdziału, aby lepiej pojąć sposób jego działania.

Montowanie partycji tylko do odczytu podczas pracy systemu

Czytelnik prawdopodobnie pamięta, że do, praktycznie rzecz biorąc, dowolnej manipulacji systemem plików należy system wymontować. Jak można przemontować partycje, nie przerywając pracy systemu? Aby na przykład przemontować partycję */home* (zakładając, że katalog ten jest umieszczony na oddzielnej partycji) w trybie tylko do odczytu, aby sprawdzić ją poleceniem `fsck`, a następnie zamontować ją z powrotem z możliwością zapisu, używamy opcji `remount` polecenia `mount`:

```
# mount -o remount,ro /home
```

Nie zadziała to, gdy zalogowany jest zwykły użytkownik, ponieważ partycja */home* będzie zajęta.

Teraz możemy bezpiecznie uruchomić na tej partycji polecenie `fsck`. Po zakończeniu przywracamy możliwość zapisu:

```
# mount -o remount,rw /home
```

Jeżeli po starcie systemu główny system plików pozostał zamontowany tylko do odczytu, nie trzeba ponownie uruchamiać systemu w celu umożliwienia zapisu — wystarczy polecenie:

```
# mount -o remount,rw /
```

Jest to łatwiejsze niż wymontowanie i ponowne zamontowanie urządzenia.

Przeglądanie zawartości dyskietek instalacyjnych

Aby zajrzeć do startowych dyskietek instalacyjnych, których obrazy są na płycie instalacyjnej Red Hata, możemy zamontować te pliki, wykorzystując urządzenia-pętle:

```
# mount -o loop /mnt/cdrom/…/plik_z_obrazem_dyskietki.img /mnt/image
```

Zaglądanie do RAM-dysku startowego

Dla zastanawiających się nad zawartością tajemniczego RAM-dysku startowego podajemy informację, że jest on zwykłym systemem plików ext2, skompresowanym przy użyciu polecenia `gzip`; rozpakujmy go w katalogu */tmp*:

```
# cp /boot/initrd-2.4.18-3.img /tmp/initrd-2.4.18-3.img.gz
# gunzip /tmp/initrd-2.4.18-3.img.gz
```

Jeżeli w systemie nie ma pliku *initrd* w katalogu */boot*, można go znaleźć na którejś z dyskietek startowych lub utworzyć przy użyciu polecenia `mkinitrd`. Teraz zamontujmy zdekompresowany obraz:

```
# mount -o loop /tmp/initrd-2.4.18-3.img /mnt/image
```

Można teraz do woli przyglądać się zawartości dysku.

Nie każdy system będzie posiadał plik *initrd*. Zwykle jest on używany do ładowania sterowników systemów plików (np. ext3, ReiserFS) lub sprzętu (chociażby kontroler Promise), które muszą być obecne, zanim system będzie mógł kontynuować procedurę startową. Niektóre dystrybucje jednodyskietkowe używają startowego RAM-dysku do załadowania minimalnego systemu operacyjnego, który z kolei dekompresuje i ładuje właściwy system plików z dyskietki.

Można w ten sposób montować także obrazy *.iso*, ale należy pamiętać, że są one zawsze tylko do odczytu ze względu na budowę systemu plików iso9660; do innych systemów plików można pisać, o ile nie zamontuje się ich tylko do odczytu. W razie potrzeby wprowadzenia zmian do pliku *.iso*, należy skopiować jego zawartość na urządzenie umożliwiające zapis, wprowadzić zmiany, a następnie utworzyć nowy obraz, korzystając z polecenia `mkisofs`.

Optymalizacja działania dysku

Wielu z nas lubi grzebać pod maską systemu, zwiększając wydajność systemu, a Linux daje nam kilka wspaniałych narzędzi do tego celu kilka wspaniałych narzędzi. Gdy moja matka powtarzała mi: „Nie naprawiaj tego, co jest popsute", mój ojciec odpowiadał: „Naprawiaj, dopóki się nie zepsuje". Zanim zabierzemy się do rzeczy, chciałbym jeszcze przedstawić dwie wskazówki, po pierwsze, przed „podkręcaniem" systemu sprawdź jego wydajność — program *bonnie* jest dobrym testem wydajności dysku twardego (nie jest częścią dystrybucji Red Hat; można go znaleźć pod adresem *http://www.coker.com.au/bonnie++/*). Po drugie, należy regulować na raz tylko jedno ustawienie, aby było wiadomo, co działa, co nie, a co jest zepsute. Niektóre z ustawień mogą nie działać lub zablokować system.

Zawsze warto mieć pod ręką sprawną dyskietkę startową; należy też pamiętać, że wszelkie optymalizacje czytelnik przeprowadza na własną odpowiedzialność. Dodatkowe wskazówki na temat optymalizacji działania systemu można znaleźć na stronie WWW o „podkręcaniu" Linuksa: *http://www.tunelinux.com/*.

Optymalizacja działania dysku w BIOS-ie i jądrze systemu

Przede wszystkim należy poszukać w dokumentacji posiadanej płyty głównej opisu dostępnych ustawień i upewnić się, że BIOS wykrywa wszystkie dyski tak, jak powinien. Zmieniać należy tylko jedno ustawienie na raz.

Sterowniki IDE jądra Linuksa znajdują się przede wszystkim w plikach *hd.c* i *ide.c*. Jeden lub oba z nich można wkompilować na stałe w jądro systemu; *ide.c* może też być modułem. Kod `hd.c` jest używany dla zgodności ze starymi kontrolerami IDE i jest domyślnie używany dla dysków na pierwszej taśmie. Na systemie z nowym kontrolerem można przeforsować użycie nowszego kodu, przekazując do jądra w linii poleceń programu startowego lub w pliku konfiguracyjnym LILO albo GRUB parametr `ide0=0x1f0`.

Inne opcje przedstawione poniżej, opracowane zostały na podstawie BootPrompt-HOWTO i dokumentacji jądra. Można ich użyć do optymalnego skonfigurowania kontrolerów i dysków IDE; zyski dla różnego sprzętu mogą się różnić, a ustawienia te nie będą działać we wszystkich systemach.

- `idex=dma` — włącza obsługę DMA.
- `idex=autotune` — próbuje dostroić interfejs w celu osiągnięcia odpowiedniej wydajności.
- `idex=ata66` — włącza obsługę dysków ATA66 (o ile dostępny jest odpowiedni kontroler).
- `hdx=ide-scsi` — włącza emulację SCSI, wymaganą dla nagrywarek CD-RW; może dawać pewną poprawę wydajności także dla zwykłych napędów CD-ROM.
- `idebus=xx` — *xx* może być dowolną liczbą od 20 do 66; wartość ta zwykle jest wykrywana automatycznie, ale jeśli w komunikatach `dmesg` znajdują się uwagi o nieudanej autodetekcji lub ustawienia zostały zmienione (przetaktowane) w BIOS-ie, można podać konkretną wartość. Większość kontrolerów PCI będzie działać z wartością 33.
- `pci=biosirq` — niektóre płyty główne działają błędnie bez tej funkcji; jeżeli opcja ta nie jest wymieniona w komunikatach `dmesg`, nie należy jej używać.

Opcje te podaje się w taki sam sposób, jak omawiane wcześniej `ide=0x1f0`.

Polecenie hdparm

Polecenie `hdparm` może zostać użyte przez superużytkownika do zmiany ustawień dysków twardych IDE (ale już nie SCSI).

Razem z łatą na jądro systemu i dodatkowymi programami obsługi dysków program ten jest częścią dystrybucji Red Hat 7.3. Eksperymentów należy dokonywać z dyskami zamontowanymi tylko do odczytu. Program działa także z napędami CD-ROM i niektórymi dyskami SCSI.

Ogólnie rzecz biorąc, sposób wywołania polecenia wygląda następująco:

```
# hdparm polecenie urządzenie
```

Strona podręcznika systemowego *hdparm* jest bardzo dokładna i zawiera szczegółowe informacje; poniżej są podane najbardziej przydatne opcje.

- `-a` — pobierz (ustaw) wskaźnik odczytu-w-przód (*read-ahead*) systemu plików. Jest to używane do zwiększania wydajności w sekwencyjnych odczytach dużych plików poprzez wstępne pobieranie dodatkowych bloków w nadziei, że okażą się one potrzebne. W obecnej wersji jądra (2.0.10) ma to domyślne ustawienie w liczbie 8 sektorów (4 KB). Wartość ta wydaje się być dobra dla większości celów, lecz w systemach, gdzie często używa się losowych przemieszczeń w pliku, lepsza może być większa wartość. Dodatkowo, wiele sterowników IDE ma także oddzielne wbudowane funkcje tego typu, które w wielu sytuacjach niwelują potrzebę odczytu-w-przód.
- `-c` — sprawdź (włącz) obsługę 32-bitowego I/O w (E)IDE. Można użyć numerycznego parametru do włączenia (wyłączenia) obsługi: `0` wyłącza obsługę 32-bitowego I/O, `1` włącza 32-bitowe transfery danych, `3` włącza 32-bitowe transfery danych ze specjalną sekwencją sync wymaganą przez wiele chipsetów. Wartość `3` działa z prawie wszystkimi chipsetami 32-bitowego IDE, lecz naraża na dodatkowe koszty. Zauważ, że „32-bit" odnosi się do transferów poprzez szynę PCI lub VLB, lecz tylko do sterownika IDE; wszystkie napędy (E)IDE mają wciąż tylko 16-bitowe połączenie ze sterownikiem.
- `-d` — włącz (wyłącz) flagę *using_dma* (użycie DMA) dla tego napędu. Opcja ta działa tylko z kilkoma kombinacjami napędów i interfejsów, które obsługują DMA, a które są znane sterownikowi IDE. Praktycznie dla szynowo zarządzanych operacji DMA z wieloma napędami obsługiwany jest chipset Intel Triton (eksperymentalnie). Dobrym pomysłem jest też używanie opcji `-X34` w połączeniu z `-d1`, aby zapewnić, że napęd sam w sobie jest zaprogramowany na drugi tryb DMA wielokrotnych słów. Używanie DMA niekoniecznie musi powodować zwiększenie wydajności, lecz wielu ludzi przy tym obstaje.
- `-i` — wyświetl informację identyfikującą, która została uzyskana od napędu podczas bootowania, o ile jest ona dostępna. Jest to właściwość nowoczesnych napędów IDE i może nie być obsługiwana przez starsze urządzenia. Zwracane dane mogą nie być aktualne, zależnie od działań od bootowania systemu. Mimo to, zawsze pokazywany jest bieżący licznik trybu wielokrotnych sektorów. Dla bardziej dokładnej interpretacji informacji identyfikującej, odsyłamy do specyfikacji *AT Attachment Interface for Disk Drives* (ANSI ASC X3T9.2 working draft, revision 4a, April 19/93).
- `-I` — żądaj informacji identyfikacji bezpośrednio od napędu. Poza tym działa podobnie do opcji `-i`.
- `-k` — pobierz (ustaw) flagę *keep_settings_over_reset* (zachowaj ustawienia po resecie). Gdy ta flaga jest ustawiona, sterownik będzie chronił opcje `-dmu` po miękkim resecie.
- `-K` — ustaw flagę *keep_features_over_reset* (zachowaj właściwości po resecie). Ustawienie powoduje, że napęd odzyskuje po miękim resecie ustawienia dla -APSWXZ. Nie wszystkie napędy obsługują tę funkcję.
- `-m` — pobierz (ustaw) licznik sektorów dla wielosektorowego wejścia-wyjścia w napędzie. Ustawienie 0 wyłącza tę funkcję. Tryb ten (inaczej znany jako IDE Block Mode) jest właściwością większości nowoczesnych dysków twardych IDE, zezwalającą na transfer wielu sektorów podczas jednego przerwania

wejścia-wyjścia zamiast tradycyjnego jednego sektora. Gdy włączona jest ta funkcja, zazwyczaj redukuje to obciążenie wejścia-wyjścia przez system o 30 – 50%. Na wielu systemach zwiększa to także przepustowość napędu w granicach od 5% do 50%. Mimo to, niektóre napędy (najbardziej zauważalnie seria WD Caviar) wydają się działać wolniej w tym trybie.

- `-p` — próba przeprogramowania chipsetu interfejsu IDE na określony tryb PIO lub próba automatycznego dostosowania się do „najlepszego" trybu PIO obsługiwanego przez napęd. Właściwość ta jest obsługiwana w jądrze tylko dla kilku „znanych" chipsetów. Niektóre chipsety IDE nie są w stanie zmienić trybu PIO dla pojedynczego napędu; w tym wypadku flaga ta może spowodować ustawienie trybu PIO dla obydwu napędów. Wiele chipsetów IDE obsługuje albo mniej, albo więcej niż standardowe 6 (od 0 do 5) trybów PIO, więc dokładne ustawienie szybkości, które właściwie jest zaimplementowane, będzie różnić się zależnie od wyrafinowania chipsetu (sterownika). Używaj z wielką ostrożnością! Funkcja ta nie daje ochrony dla nieuważnych, a niepomyślne działanie może spowodować poważne uszkodzenie systemu plików!

- `-q` — obsłuż następną flagę cicho, nie wydając komunikatów na wyjście. Jest to użyteczne dla redukowania zamieszania na ekranie w wypadku uruchamiania z */etc/rc.d/rc.local*. Nie stosuje się do flag `-i`, `-v`, `-t` i `-T`.

- `-r` — pobierz (ustaw) flagę *read-only* (tylko do odczytu) urządzenia. Gdy jest ustawiona, operacje zapisu nie są na tym urządzeniu dozwolone.

- `-T` — dokonaj pomiarów czasu odczytów cache dla celów porównawczych i testów wydajnościowych. Aby uzyskać znaczące wyniki, operacja ta powinna być powtarzana 2-3 razy na nieaktywnym pod innymi względami systemie (bez innych aktywnych procesów) z przynajmniej kilkoma megabajtami wolnej pamięci. Wyświetla szybkość odczytu bezpośrednio z linuksowej pamięci podręcznej buforów dyskowych, bez dostępu do dysku. Ta miara jest wskaźnikiem przepływu danych między procesorem, pamięcią podręczną i pamięcią systemu. Jeśli podano również flagę `-t`, to wskaźnik poprawności, oparty na wyniku `-T`, zostanie włączony do wyniku zgłaszanego przez operację `-t`.

- `-t` — dokonaj pomiarów czasu odczytów z urządzenia dla celów porównawczych i testów wydajnościowych. Aby uzyskać znaczące wyniki, operacja ta powinna być powtarzana 2-3 razy na nieaktywnym pod innymi względami systemie (bez innych aktywnych procesów) z przynajmniej kilkoma megabajtami wolnej pamięci. Wyświetla to szybkość odczytu poprzez cache buforowy z dysku, bez wcześniejszego cache'owania danych. Ta miara jest wskaźnikiem tego, jak szybko napęd jest w stanie obsługiwać sekwencyjne odczyty danych pod Linuksem bez obciążenia związanego z systemem plików. Aby zapewnić dokładne pomiary, cache jest wypróżniany przy użyciu ioctl `BLKFLSBUF`.

- `-u` — pobierz (ustaw) flagę *interrupt-unmask* napędu. Ustawienie `1` zezwala sterownikowi na wyłączenie maskowania innych przerwań podczas przetwarzania przerwania dyskowego, co poprawia w dużej mierze płynność działania systemu i eliminuje błędy „serial port overrun". Używaj tej funkcji z ostrożnością: niektóre kombinacje napęd-kontroler nie radzą sobie z nią dobrze, co może spowodować poważne uszkodzenie systemu plików. W szczególności interfejsy CMD-640B i RZ1000 (E)IDE mogą być niestabilne (z powodu usterek sprzętowych),

gdy opcja ta jest używana z wersjami jądra wcześniejszymi niż 2.0.13. Wyłączenie funkcji *IDE prefetch* tych interfejsów (zwykle ustawienie BIOS/CMOS) daje bezpieczne rozwiązanie tego problemu dla wcześniejszych jąder.

- `-W` — włącz (wyłącz) pamięć podręczną zapisu w napędach IDE (domyślnie wyłączone).
- `-X` — ustaw tryb transferu IDE dla nowszych napędów (E)IDE/ATA2. Jest to zazwyczaj używane w połączeniu z `-d1`, gdy włącza się DMA do (z) napędu na obsługiwanym układzie kontrolera (jak na przykład Intel 430FX Triton) przy użyciu `-X34` do wyboru trybu transferów multiword DMA mode2. Poza tym użycie tej flagi jest rzadko potrzebne, gdyż większość (wszystkie) nowoczesnych napędów IDE domyślnie ustawia się na najszybszy ze swoich trybów PIO przy włączaniu. Zmiana tego zachowania może być bezużyteczna i ryzykowna. Na dyskach, które wspierają alternatywne tryby transferu, można użyć `-X` do przełączania samego trybu napędu. Przed zmianą trybu transferu interfejs IDE powinien być uzworkowany lub zaprogramowany (zobacz flagę `-p`) dla nowego trybu, aby zapobiec utracie i (lub) zniszczeniu danych. Używaj tego ze szczególną uwagą! Dla trybów transferu PIO (*Programmed Input/Output*) używanych przez Linuksa wartość ta jest po prostu oczekiwanym numerem trybu PIO plus 8. Wartość 09 ustawia więc tryb *PIO mode1*, 10 *PIO mode2*, a 11 wybiera *PIO mode3*. Ustawienie 00 odtwarza „domyślny" tryb PIO dysku, a 01 wyłącza IORDY.

Optymalizacja działania systemów plików

Nie ma sensu zostawianie rzeczy takimi, jakie są. Linux dostarcza kilku narzędzi do dostrajania ustawień systemu plików. Uważa się, że producenci sprzętu i twórcy dystrybucji mają tendencje do wybierania raczej konserwatywnych ustawień, które będą zawsze działać, nie wykorzystując jednak pełnego potencjału systemu, w czym niniejsza książka usiłuje pomóc.

Projektanci systemu plików Linuksa bardzo dobrze wybrali domyślne wartości używane przy tworzeniu systemu plików. Mimo że działają one dobrze dla większości użytkowników, w niektórych zastosowaniach serwerowych można zyskać na wydajności po zmianie pewnych parametrów systemu plików. Jak zwykle, sprawdzaj działanie wprowadzonych zmian za pomocą testów wydajnościowych.

Polecenie mke2fs

Opcja `-O sparse_super` tworzy mniejszą ilość kopii superbloku, pozostawiając większą ilość miejsca do wykorzystania na pliki. Nie poprawia to w zasadzie wydajności, ale na dużych dyskach może zwolnić dodatkowe miejsce. Opcja jest domyślnie włączona dla jąder systemu serii 2.2 i późniejszych; jest tu wymieniona tylko dlatego, że wiele źródeł nie zwraca na ten fakt uwagi (opcję tę można zmienić w utworzonym już systemie plików za pomocą polecenia `tune2fs`).

Można też ustalić rozmiar bloku przy użyciu opcji `-b rozmiar`. Może on mieć wpływ na wydajność systemu plików, bo duże bloki lepiej działają dla dużych plików i *vice versa*. Zasadniczo nie ma reguł dotyczących rozmiaru bloku i zaleca się pozostanie przy domyślnych 1024 bajtach lub spędzenie sporej ilości czasu na testowaniu wydajności przy różnych ustawieniach.

Polecenie tune2fs

Używając polecenia `tune2fs`, można dostroić kilka zmiennych parametrów systemów ext2 lub ext3. Oto kilka związanych z wydajnością.

- Aby wyłączyć okresowe sprawdzanie systemu plików, ustaw wartość licznika montowań przy której system będzie sprawdzany, na zero (opcja `-c 0`).
- Odstęp czasowy między kolejnymi wymuszanymi sprawdzeniami systemu plików ustala się opcją `-i`.
- Opcja `-m` może ustalić mniejszą ilość bloków zarezerwowanych dla superużytkownika (podaje się wartość w procentach), uwalniając miejsce dla użytkowników kosztem przestrzeni na odzyskane pliki dla `fsck`.
- Można zmniejszyć ilość kopii superbloku przy użyciu opcji `-O sparse_super` (nowo tworzone systemy plików mają to ustawione domyślnie). Po ustawieniu tej opcji należy uruchomić `e2fsck`.
- Jeszcze więcej przestrzeni można zwolnić, posługując się poleceniem `-r`, ustawiającym ilość bloków zarezerwowanych dla superużytkownika.

Warto zauważyć, że większość tych opcji zwalnia część miejsca na dysku kosztem możliwości odzyskania danych przez `fsck`. O ile nie ma konieczności wykorzystania każdego dostępnego fragmentu dysku bez względu na konsekwencje, należy pozostać przy wartościach domyślnych — duże dyski są coraz tańsze.

Polecenie e2fsck

Polecenie to sprawdza zarówno systemy ext3, jak i ext2. Oto niektóre z przydatniejszych opcji, według strony podręcznika systemowego.

- `-b superblok` — zamiast użyć domyślnego superbloku, użyj podanej w argumencie kopii zapasowej. Ta opcja jest zwykle używana w razie zniszczenia domyślnego superbloku. Położenie superbloku zapasowego zależy od rozmiaru bloku systemu plików; dla systemów z jednokilobajtowymi blokami superblok można znaleźć w bloku 8193; przy dwukilobajtowych blokach będzie on w bloku 16384, a przy czterokilobajtowych — w bloku 32768.

Można określać dodatkowe superbloki, wywołując polecenie `mke2fs` z opcją `-n`, powodującą wypisanie położenia superbloków. Aby polecenie wypisało prawidłowe wartości, należy użyć opcji `-b`, podając rozmiar bloku.

Jeżeli podano superblok, a system nie został otwarty tylko do odczytu, `e2fsck` zapisze w głównym superbloku prawidłowe informacje na zakończenie sprawdzenia systemu.

- `-c` — włącza poszukiwanie uszkodzonych sektorów dysku przy użyciu polecenia `badblocks` i oznaczanie ich przez dodanie numerów bloków do i-węzła uszkodzonych bloków.
- `-f` — wymusza sprawdzenie, nawet gdy system wydaje się czysty.
- `-l` *`plik`* — dodaj bloki o numerach podanych w pliku do listy uszkodzonych bloków. Format pliku jest taki sam, jak tworzony przez polecenie `badblocks`.
- `-L` *`plik`* — ustawia listę uszkodzonych bloków na zawartość podanego pliku. Opcja działa analogicznie do `-l`, z tym, że przed dodaniem zawartości pliku lista uszkodzonych bloków jest czyszczona.
- `-n` — otwiera system plików tylko do odczytu, zakładając odpowiedź „nie” na wszystkie pytania. Umożliwia nieinteraktywne użycie `e2fsck`.
- `-p` — automatycznie naprawia system plików bez zadawania pytań.
- `-v` — włącza szczegółowy opis działań.

Polecenie badblocks

Nie jest programem do dostrajania wydajności, `badblocks` sprawdza partycję, najlepiej wymontowaną, w poszukiwaniu uszkodzonych sektorów dysku. Oto niektóre bardziej przydatne opcje wymienione na stronie podręcznika systemowego.

- `-f` — normalnie `badblocks` odmówi wykonania testu z zapisem na zamontowanym systemie plików, ponieważ mogłoby to doprowadzić do zapaści systemu. Można zmusić program do przeprowadzenia testu przy użyciu opcji `-f`, ale w normalnych warunkach nie powinno się tego robić. Jedynym przypadkiem, gdy opcja ta jest bezpieczna w użyciu, jest nieprawidłowa zawartość pliku */etc/mtab*, wskazująca na zamontowanie partycji, gdy faktycznie nie jest ona zamontowana.
- `-i` *`plik_wejściowy`* — czyta z pliku listę znanych już uszkodzonych bloków. Program nie będzie wtedy sprawdzał tych bloków, ponieważ wiadomo, że są one uszkodzone. Gdy zamiast nazwy pliku poda się „-”, lista będzie odczytana ze strumienia wejściowego. Bloki wymienione w tym pliku nie będą zapisane na ekran ani do pliku wyjściowego. Można użyć opcji `-b` polecenia `dumpe2fs`, aby uzyskać spis bloków oznaczonych jako uszkodzone w formacie odpowiednim do użycia w tej opcji.
- `-o` *`plik_wyjściowy`* — wypisz listę uszkodzonych bloków do podanego pliku. Bez tej opcji program wyświetla numery bloków na ekranie. Format pliku jest odpowiedni do wykorzystania z opcją `-l` poleceń `e2fsck` i `mke2fs`.
- `-n` — wykonuje niedestruktywny test z zapisem; domyślnie wykonywany jest jedynie niedestruktywny test z samym odczytem. Nie można łączyć tej opcji z opcją `-w`, ponieważ wykluczają się one wzajemnie.

- `-s` — wyświetlaj postęp operacji, wypisując numery sprawdzanych w danej chwili bloków.
- `-v` — włącza szczegółowy opis działań.
- `-w` — wykonuje test z zapisem. W tym teście `badblocks` szuka uszkodzonych bloków, zapisując pewne wzorce (`0xAA`, `0x55`, `0xFF`, `0x00`) na każdym bloku urządzenia, następnie odczytując blok i porównując zawartość. Nie można używać tej opcji łącznie z `-n`.

Opcja montowania noatime

Opcji tej używa się w pliku *fstab*; wyłącza ona zapisywanie znacznika czasu ostatniego dostępu do i-węzła. Daje to wzrost prędkości działania systemu plików, gdy jest on obciążony częstym dostępem, a informacja o czasie ostatniego dostępu nie jest istotna.

Zasoby

- *http://www.linuxdoc.org/HOWTO/Filesystems-HOWTO.html* — w dokumencie Filesystems-HOWTO można znaleźć wiele informacji zarówno na temat własnych systemów plików Linuksa, jak i na temat systemów bardziej egzotycznych.
- *http://www.linuxdoc.org/HOWTO/mini/Partition/index.html* — wiele szczegółowych informacji na temat partycjonowania dysku.
- *http://people.spoiled.org/jha/ext3-faq.html* — nieoficjalne FAQ linuksowego systemu plików ext3.
- *http://www.linuxdoc.org/HOWTO/mini/Ext2fs-Undeletion.html* — dokument ten może pomóc w odzyskaniu skasowanego właśnie z partycji ext2/ext3 pliku.
- *http://www.linuxdoc.org/HOWTO/mini/Ext2fs-Undeletion-Dir-Struct/index.html* — nawet w przypadku skasowaniu całego katalogu są jeszcze szanse odzyskania danych.
- *http://www.linuxdoc.org/HOWTO/mini/Loopback-Root-FS.html* — możemy korzystać nie tylko ze zwykłego systemu plików, ale też z głównego systemu plików zawartego w pliku zapisanego w innym systemie plików; sposób wykonania takiej konfiguracji znajduje się w tym dokumencie.
- *http://www.linuxdoc.org/HOWTO/Loopback-Encrypted-Filesystem-HOWTO.html* — dokument opisujący konfigurację i używanie szyfrowanego systemu plików.
- *http://www.linuxdoc.org/HOWTO/LVM-HOWTO.html* — czas pożegnać się z przestarzałymi dyskami fizycznymi i systemami plików o ograniczonym rozmiarze; pomoże w tym z pewnością ten dokument, opisujący system LVM.
- *http://www.linuxdoc.org/HOWTO/UMSDOS-HOWTO.html* — dokument zawierający informacje na temat korzystania z systemu plików umsdos.

- *http://www.linuxdoc.org/HOWTO/mini/NFS-Root.html*, *http://www.linuxdoc.org/HOWTO/mini/NFS-Root-Client-mini-HOWTO/index.html* — opis konfiguracji i używania głównego systemu plików montowanego zdalnie jako dysk NFS.
- *http://www.nyx.net/~sgjoen/disk.html* — HOWTO na temat podkręcania wydajności systemu z wieloma dyskami; zawiera także ogólne informacje o dyskach, kontrolerach i systemach plików.
- *http://www.linuxdoc.org/HOWTO/Software-RAID-HOWTO.html* — omówienie konfiguracji programowych macierzy RAID.
- *http://www.linuxdoc.org/HOWTO/mini/DPT-Hardware-RAID.html* — omówienie konfiguracji sprzętowej macierzy RAID, z naciskiem na macierze produkowane przez DPT.
- *http://www.linuxdoc.org/HOWTO/Tips-HOWTO.html* — ogólne wskazówki na temat używania Linuksa, w tym odpowiedzi na pytania dotyczące systemów plików, takie jak „czy mam wystarczająco dużo wolnej przestrzeni?” albo „jak przenosić katalogi między różnymi systemami plików?”.
- *http://www.linuxdoc.org/HOWTO/Large-Disk-HOWTO.html* — dokument dokładnie omawiający wszelkie kłopoty związane z używaniem dużych dysków i sposoby ich rozwiązania.
- *http://www.linuxdoc.org/HOWTO/BootPrompt-HOWTO.html* — dokument ten omawia przekazywanie argumentów do jądra Linuksa podczas startu systemu, włącznie ze spisem najpotrzebniejszych argumentów.
- *http://www.atnf.csiro.au/~rgooch/linux/docs/devfs.html* — FAQ linuksowego systemu plików urządzeń Devfs.
- *http://www.linux-usb.org/USB-fuide/x498.html* — rozdział linuksowego przewodnika po urządzeniach USB dotyczący dysków.
- *http://www.tunelinux.com/* — strona z informacjami na temat optymalizacji systemu Linux.
- *http://www.coker.com.au/bonnie++/* — strona narzędzia do pomiaru wydajności dysków *bonnie*.