

# Administracja systemami uniksowymi dla zastosowań naukowych i komercyjnych

Tom I

Krzysztof Boryczko

# Spis treści

<b>1</b>	<b>Wprowadzenie</b>	<b>5</b>
1.1	Definicja i klasyfikacja systemów operacyjnych . . . . .	5
1.2	Krótki rys historyczny . . . . .	7
1.3	Architektura systemu Unix . . . . .	9
1.3.1	Budowa ogólna . . . . .	9
1.3.2	Architektura jądra . . . . .	11
1.4	Zalety i wady systemu . . . . .	13
1.5	Konwencje przyjęte w kolejnych rozdziałach . . . . .	13
<b>2</b>	<b>Użytkownicy i grupy użytkowników w systemie Unix</b>	<b>15</b>
2.1	Podstawowe pliki konfiguracyjne w dystrybucji Fedora . . . . .	15
2.1.1	Plik <i>/etc/passwd</i> . . . . .	15
2.1.2	Plik <i>/etc/shadow</i> . . . . .	17
2.1.3	Plik <i>/etc/group</i> . . . . .	19
2.1.4	Plik <i>/etc/gshadow</i> . . . . .	21
2.2	Zarządzanie użytkownikami w dystrybucji Fedora . . . . .	21
2.2.1	Dodawanie użytkownika do systemu . . . . .	22
2.2.2	Zmiana konfiguracji użytkownika . . . . .	26
2.2.3	Monitorowanie podłączania się użytkowników do systemu . . . . .	32
2.2.4	Ograniczanie dostępu do systemu . . . . .	38
2.2.5	Usuwanie konta użytkownika . . . . .	42
2.3	Zarządzanie grupami użytkowników w dystrybucji Fedora . . . . .	43
2.3.1	Dodawanie grupy użytkowników . . . . .	43
2.3.2	Zmiana konfiguracji grupy . . . . .	44
2.3.3	Usuwanie grupy użytkowników . . . . .	46
2.4	Spójność plików konfiguracyjnych w dystrybucji Fedora . . . . .	46
2.4.1	Pliki konfiguracyjne użytkowników . . . . .	47
2.4.2	Pliki konfiguracyjne grup . . . . .	49
2.5	Podstawowe pliki konfiguracyjne w dystrybucji FreeBSD . . . . .	50
2.5.1	Plik <i>/etc/master.passwd</i> . . . . .	50
2.5.2	Plik <i>/etc/passwd</i> . . . . .	51
2.5.3	Plik <i>/etc/pwd.db</i> . . . . .	52
2.5.4	Plik <i>/etc/group</i> . . . . .	52
2.6	Zarządzanie użytkownikami w systemie FreeBSD . . . . .	53
2.6.1	Dodawanie użytkownika do systemu . . . . .	53
2.6.2	Zmiana konfiguracji użytkownika . . . . .	59
2.6.3	Monitorowanie podłączania się do systemu . . . . .	66

2.6.4	Ograniczanie dostępu do systemu . . . . .	73
2.6.5	Usuwanie użytkownika z systemu . . . . .	74
2.7	Zarządzanie grupami użytkowników w dystrybucji FreeBSD . . . . .	76
2.7.1	Dodawanie grupy użytkowników . . . . .	76
2.7.2	Zmiana konfiguracji grupy . . . . .	77
2.7.3	Usuwanie grupy użytkowników . . . . .	78
2.8	Spójność plików konfiguracyjnych w dystrybucji FreeBSD . . . . .	79
2.8.1	Pliki konfiguracyjne użytkowników . . . . .	79
2.8.2	Pliki konfiguracyjne grup . . . . .	80
2.9	Pluggable Authentication Modules (PAM) . . . . .	81
2.9.1	Konfiguracja . . . . .	81
2.9.2	Podstawowe moduły . . . . .	85
2.9.3	Przykłady zastosowań . . . . .	89
2.10	Ograniczanie zasobów udostępnianych użytkownikom . . . . .	93
2.10.1	Mechanizmy dostępne w dystrybucji Fedora . . . . .	93
2.10.2	Mechanizmy dostępne w dystrybucji FreeBSD . . . . .	96
<b>3</b>	<b>Dyskowe systemy plików</b>	<b>107</b>
3.1	Przegląd podsystemu plików . . . . .	107
3.2	Informacja przechowywana w i-węźle . . . . .	111
3.2.1	Typ pliku . . . . .	112
3.2.2	Prawa dostępu . . . . .	117
3.2.3	Właściciel indywidualny i grupowy . . . . .	134
3.2.4	Rozmieszczenie pliku w systemie plików . . . . .	136
3.3	Podstawy administracji systemami plików . . . . .	138
3.3.1	Podział drzewa katalogów pomiędzy systemy plików . . . . .	141
3.3.2	Montowanie systemu plików . . . . .	143
3.3.3	Odmontowanie systemu plików . . . . .	155
3.3.4	Reglamentowanie zasobów systemu plików - kontyngenty dyskowe (quota) . . . . .	161
3.3.5	Dziennikowane systemy plików . . . . .	185
3.4	Administracja wybranymi systemami plików w dystrybucji RedHat . . . . .	188
3.4.1	Nazwy plików urządzeń w dystrybucji RedHat . . . . .	188
3.4.2	Zarządzanie partycjami . . . . .	190
3.4.3	System plików ext3 . . . . .	197
3.4.4	System plików ext4 . . . . .	205
3.4.5	System plików XFS . . . . .	221
3.4.6	Journalized File System (JFS) . . . . .	233
3.4.7	System plików ReiserFS . . . . .	248
3.4.8	System plików Btrfs (B-tree file system) . . . . .	279
3.5	Administracja systemami plików w dystrybucji FreeBSD . . . . .	291
3.5.1	Nazwy plików urządzeń w dystrybucji FreeBSD . . . . .	291
3.5.2	Zarządzanie paskami i partycjami . . . . .	292
3.5.3	System plików UFS . . . . .	301
3.5.4	System plików ZFS (Z file system) . . . . .	317
3.5.5	Systemy plików z rodziny ext w systemie FreeBSD . . . . .	320
3.6	Logical Volume Manager (LVM) w dystrybucjach RedHat . . . . .	320
3.7	Programowe macierze dyskowe . . . . .	340
3.7.1	Podstawy teoretyczne . . . . .	341
3.7.2	Programowe macierze dyskowe w dystrybucji RedHat . . . . .	353

3.7.3	Organizacja pamięci masowej w dystrybucji FreeBSD . . . . .	369
<b>4</b>	<b>Procesy</b>	<b>374</b>
4.1	Wprowadzenie . . . . .	374
4.1.1	Struktury opisujące proces w systemie operacyjnym . . . . .	376
4.1.2	Stany procesu . . . . .	378
4.1.3	Szeregowanie zadań . . . . .	380
4.2	Plik <i>/etc/inittab</i> . . . . .	386
4.3	Crontab . . . . .	386
4.4	Obsługa plików dziennika . . . . .	386
4.4.1	Rejestrowanie zdarzeń w dystrybucji Fedora . . . . .	386
4.4.2	Rejestrowanie zdarzeń w dystrybucji FreeBSD . . . . .	386
<b>5</b>	<b>Interpretery poleceń</b>	<b>389</b>
5.1	Systematyka . . . . .	389
5.2	Działanie . . . . .	390
5.3	Konfiguracja . . . . .	394
5.3.1	Zmienne specjalne . . . . .	395
5.3.2	Zmienne środowiska . . . . .	395
5.3.3	Zmienne programowe . . . . .	395
5.4	Programowanie w językach interpreterów poleceń . . . . .	395
5.4.1	Polecenie <i>expr</i> . . . . .	395
5.4.2	Polecenie <i>test</i> . . . . .	398
5.4.3	Zasady uruchamiania skryptów . . . . .	400
5.4.4	Przykładowe, proste skrypty . . . . .	400
5.4.5	Przykładowe skrypty . . . . .	403
<b>6</b>	<b>Zarządzanie oprogramowaniem</b>	<b>406</b>
<b>7</b>	<b>Podstawy konfiguracji sieciowych</b>	<b>407</b>
<b>8</b>	<b>Badanie wydajności systemu i optymalizacja</b>	<b>408</b>

# Rozdział 1

## Wprowadzenie

Powstanie systemu operacyjnego Unix datuje się na rok 1969. Zyskał on znaczną popularność przede wszystkim ze względu na jego dostępność na szerokiej klasie komputerów, od osobistych po superkomputery. Na każdym z nich zapewnia jednakowe środowisko pracy. Z dobrym przybliżeniem można powiedzieć, że składa się z dwóch części. Pierwsza to jądro systemu operacyjnego, które zapewnia komunikację ze sprzętem. Drugą stanowią usługi i programy użytkowe, które uczyniły system Unix tak popularnym.

W rozdziale tym omówiono kilka podstawowych pojęć z teorii systemów operacyjnych oraz przytoczono ich klasyfikację. Pojawił się krótki rys historyczny systemu operacyjnego Unix, przedstawiono ogólną strukturę jego budowy, zwrócono uwagę na zalety i wady. Na zakończenie określono konwencje obowiązujące w dalszych rozdziałach.

### 1.1 Definicja i klasyfikacja systemów operacyjnych

W literaturze można spotkać dwie uzupełniające się definicje systemu operacyjnego. Pierwsza z nich [1] mówi, że **system operacyjny to program działający jako pośrednik między użytkownikiem komputera, a sprzętem komputerowym**. System operacyjny ma tworzyć środowisko, w którym użytkownik będzie wykonywał programy. Podstawowym wymaganiem stawianym systemowi operacyjnemu jest spowodowanie, aby stworzony z jego udziałem system komputerowy był wygodny w użyciu. Na drugim miejscu stawia się wydajną eksploatację sprzętu. Na aspekty spełnienia tego postulatu będziemy często zwracali uwagę.

Druga definicja mówi, że system operacyjny to **zarządca zasobów**. W systemie komputerowym wyróżniamy dwa rodzaje zasobów. Zasoby bierne, to zasoby sprzętowe, a więc procesor jeden lub wiele zarówno jedni, jak i woelordzeniowy, pamięć operacyjna, pamięć masowa, itd. Zasobami czynnymi są wykonujące się w nim procesy. Procesy do wykonywania się (niektórzy mówią nawet życia) potrzebują zasobów biernych. Rolą systemu operacyjnego jest przydzielanie im tych zasobów. Jeśli w systemie komputerowym uruchomiony jest tylko jeden proces, to działanie systemu operacyjnego sprowadza się do przydzielania mu całych zasobów występujących fizycznie w tym systemie. Jeśli natomiast uruchomionych zostało kilka procesów, to przydzielanie im zasobów w sposób uwzględniający spełnienie postulatów pierwszej definicji nie jest zadaniem trywialnym. Szczególnie dotyczy to najistotniejszego zasobu biernego jakim jest procesor. To jego wydajność i to jak dużo czasu spędza na wykonywaniu instrukcji naszych procesów decyduje o naszym zadowoleniu z systemu komputerowego.

Historycznie pierwszymi systemami operacyjnymi były tak zwane systemy wsadowe. Wykonywały one kolejno zadania ze zbioru (wsadu) w kolejności takiej, jak zadania te zostały we

wsadzie umieszczone. Jeśli, w trakcie wykonania zadanie wykonywało operację wejścia/wyjścia, procesor czekał na jej zakończenie. Należy w tym miejscu zwrócić uwagę, że operacje komunikacji ze światem zewnętrznym wykonywane przez procesy są dla systemu operacyjnego najbardziej uciążliwe. Współczesne procesory są taktowane zegarami o częstotliwościach rzędu kilku  $GHz$ . W ciągu sekundy przetwarzają miliardy ( $10^9$ ) bajtów. Stąd każda sekundna bezczynności procesora może prowadzić do nie wykonania wielu zadań. Przy takim podejściu do zarządzania zadaniami postulat, dotyczący wydajnej eksploatacji sprzętu przez wsadowy system operacyjny nie jest spełniony.

Wieloprogramowe systemy wsadowe stanowiły kolejny etap rozwoju systemów operacyjnych. Pojawiło się pojęcie puli zadań, z której zadania mogły być wybierane do wykonania. W systemach tych zastosowano proste algorytmy planowania wykonania zadań. Aby procesor nie był bezczynny w trakcie wykonywania przez proces operacji wejścia/wyjścia zastosowano proste rozwiązanie polegające na przejściu do wykonania instrukcji kolejnego procesu. Jest to możliwe, gdyż operacje wejścia/wyjścia mają charakter „zamawiany”, tzn. procesor zleca ich wykonanie odpowiednim urządzeniom, a one sygnalizują zakończenie ich wykonania przerwaniem. Stąd w pewnym momencie pierwszy proces zakończy operację wejścia/wyjścia i otrzyma procesor do wykonywania dalszych instrukcji. Zauważmy, że procesor nie będzie bezczynny dopóty, dopóki w puli jest choć jedno zadanie do wykonania.

Logicznym rozszerzeniem wieloprogramowości jest podział czasu. Procesor wykonuje na przemian wiele różnych zadań, przy czym przekazywanie procesora kolejnym procesom występuje na tyle często, że użytkownik ma wrażenie, iż wykonują się one jednocześnie. Dodatkowo ma możliwość współdziałania z każdym programem podczas jego wykonania. Systemy operacyjne działające z podziałem czasu umożliwiają zatem budowanie interaktywnych systemów komputerowych. Zagadnienie efektywnego wykorzystania procesora zostało rozwiązane podobnie jak w systemach wieloprogramowych. Operacje wejścia/wyjścia są zlecane odpowiednim urządzeniom, a proces czeka na ich zakończenie. W tym czasie procesor może wykonywać instrukcje innych procesów. Jeśli natomiast chodzi o kryteria przydziału procesora, to istotne stają się dwa zagadnienia. Po pierwsze na jak długo przydzielać procesor danemu procesowi i po drugie w jakiej kolejności procesor ma być przydzielany procesom. Rozwiązania pierwszego zagadnienia doprowadziły do pojawienia się systemów operacyjnych tzw. kooperacyjnych, w których procesor był przydzielany procesowi do wykonania określonej liczby instrukcji. Algorytm przydziału określał wartość licznika rozkazów, przy której procesor był zabierany danemu procesowi. Podejście takie mogło prowadzić do zawłaszczenia procesora przez proces. Miało to miejsce na przykład wtedy, gdy w obrębie instrukcji do wykonania pojawiła się pętla nieskończona. System śledził wówczas zmiany wartości licznika rozkazów oczekując na pojawienie się tej wyznaczonej, co nigdy nie następowało. Współczesne systemy operacyjne pracują w oparciu o wywłaszczanie. Technika ta polega na przydzielaniu procesora danemu procesowi na ściśle określony przedział czasu, zwany kwantem. Jego wartość jest albo stała i jednakowa dla każdego zadania, albo zależna np. od priorytetu zadania. Po wykorzystaniu przez proces przydzielonego mu kwantu czasu procesor był przekazywany następnemu procesowi. Przy takim podejściu, pojawienie się w instrukcjach procesu pętli nieskończonej nie doprowadzi do zawłaszczenia procesora, gdyż będzie ona wykonywana tylko w tych przedziałach czasu, w których procesor będzie przydzielony danemu procesowi. Poza nimi procesor będzie wykonywał instrukcje innych procesów. W jakiej kolejności przydzielać procesor procesom? Rozwiązań jest wiele. Najczęściej stosowane polegają na wyznaczeniu wartości priorytetu procesu będącego miarą jego ważności w systemie. Dokonuje się tego w oparciu o wiele czynników, jak np. ilość zużytego czasu procesora, liczba i częstość wykonywania operacji wejścia/wyjścia, itd. Następnie procesor jest przydzielany procesom od najważniejszego począwszy.

Reasumując, wymagania stawiane systemom operacyjnym doprowadziły do powstania syste-

mów wielozadaniowych pracujących w trybie z wywłaszczaniem. Potencjalnie dają one największe możliwości wykorzystania mocy obliczeniowej sprzętu komputerowego. Wymagają jednak dokładnego poznania mechanizmów działania oraz możliwości ich konfiguracji.

## 1.2 Krótki rys historyczny

Początki systemu operacyjnego Unix sięgają roku 1965. Wówczas w Bell Telephone Laboratories, General Electric Company oraz Massachusetts Institute of Technology podjęto próby stworzenia systemu operacyjnego o nazwie Multics [2]. Miał on zapewnić równoczesny dostęp do komputera wielu użytkownikom, dostarczać moc obliczeniową, zasoby pamięci do przechowywania danych oraz umożliwiać ich współdzielenie. W założeniach miał więc być systemem wieloużytkownikowym oraz wielozadaniowym. Prosta wersja systemu Multics działała na komputerze GECOS Honeywell GE 635 już przed rokiem 1969. Nie spełniała ona jednak założeń projektowych. Co gorsza nikt nie potrafił powiedzieć, kiedy pojawi się wersja produkcyjna systemu. Było to powodem odstąpienia pracowników Bell Laboratories od projektu, w wyniku czego pozostali oni bez wygodnego środowiska do prowadzenia obliczeń interakcyjnych.

W tym czasie Ken Thompson, Dennis Ritchie i inni podjęli próbę implementacji systemu operacyjnego. Pracę rozpoczęto od projektu i implementacji systemu plików, który przekształcił się we wczesną wersję systemu plików systemu Unix. Ken Thompson napisał następnie symulatory jego działania, zaimplementował pierwowzór pamięci wirtualnej z mechanizmem stronicowania na żądanie oraz stworzył jądro systemu dla komputera GE 645. System komputerowy okazał się jednak kłopotliwy i kosztowny w eksploatacji, głównie ze względu na ograniczenia sprzętowe. Dlatego przeniesiono go na komputer PDP-7. Implementacja ta była wyposażona we wczesną wersję uniksowego systemu plików, posiadała podsystem obsługi procesów oraz małą liczbę programów usługowych. System otrzymał nazwę Unix – kalambur, którego źródłem była nazwa systemu Multics, a autorem Brian Kernigham.

Zbudowany system operacyjny znalazł zastosowanie w systemie przetwarzania tekstu wydziału patentowego Bell Laboratories. W roku 1971 system zostaje przeniesiony na komputer PDP-11. Powstają kolejne programy usługowe. Prace Kena Thompsona nad implementacją kompilatora języka FORTRAN owocują projektem języka B oraz implementacją jego interpretera. Język stworzony z myślą o implementowaniu systemów operacyjnych, ze względu na to, iż był językiem interpretacyjnym, posiadał dość duże braki wydajnościowe. Dlatego Dennis Ritchie rozwinął go w język C, który posiadał kompilator generujący kod maszynowy. Sam język umożliwiał deklarowanie typów danych, tworzenie struktur danych oraz wyposażony był liczne operacje na pamięci i plikach. W roku 1973 system Unix zaimplementowano na nowo, wykorzystując w tym celu język C. Liczba pracujących instalacji w Bell Laboratories wynosiła wówczas 25.

W roku 1974 Ken Thompson i Dennis Ritchie opublikowali artykuł opisujący system operacyjny Unix w *Communication of the ACM* [3]. Była to forma promowania nowego systemu. W tym czasie AT&T na mocy porozumienia podpisanego z rządem federalnym USA nie mogło handlować systemami komputerowymi. Dlatego Unix był przekazywany uniwersytetom wykorzystującym go do celów dydaktycznych. Zyskał on również popularność w firmach telefonicznych, gdyż dostarczał usługi realizujące transakcje sieciowe w czasie rzeczywistym oraz zapewniał elastyczne środowisko do konstruowania oprogramowania. Unix znalazł też zastosowanie w pracach biurowych. W roku 1977 firma Interactive Systems Corporation zaczęła dystrybuować system Unix wyposażony w programy narzędziowe służące właśnie automatyzacji prac biurowych.

Kolejne lata przyniosły wzrost popularności mikroprocesorów oraz pojawienie się na rynku wielu różnorodnych architektur komputerowych. Produkujące je firmy implementowały swoje własne wersje systemu Unix, wzbogacając je często o własne modyfikacje i usprawnienia. Spo-

wodowało to pojawienie się kilku wariantów systemu podstawowego. W latach 1977–1982 w Bell Laboratories połączono kilka wersji pochodzących z AT&T w jeden system, znany później jako Unix System III. Po wprowadzeniu kilku istotnych zmian, zmodyfikowany system nazwano Unix System V. Wersja IV systemu była wewnętrzną - nie ujrzała światła dziennego. W roku 1983 pojawiła się pierwsza komercyjna wersja systemu z AT&T – System V. Wraz z komercjalizacją AT&T przestała udostępniać kod źródłowy systemu poza licencjami komercyjnymi. Spowodowało to protesty wśród inżynierów oraz pracowników akademickich wielu uniwersytetów i doprowadziło do powstania na bazie społeczności użytkowników i niezależnych twórców Uniksa ruchu *wolnego oprogramowania*. Założona w roku 1983 Free Software Foundation (FSF)<sup>1</sup> postawiła sobie za cel stworzenie wolnego systemu uniksowego bez kodu pochodzącego z AT&T. W roku 1988 firmy Sun oraz AT&T zainicjowały porozumienie, którego wynikiem było powstanie konsorcjum Unix International oraz wzorcowego systemu System V Release 4 (SVR4) łączącego zalety Systemu V oraz wersji BSD. Wynikiem porozumienia było także powstanie nowego systemu firmy Sun – Solarisa. Konkurencyjni wobec porozumienia producenci zawiązali Open Software Foundation (OSF)<sup>2</sup>, która wespół z systemem OSF/1 silnie bazujący na wersji BSD. W 1988 roku opublikowano specyfikację POSIX.1<sup>3</sup> stanowiącym podstawę tego, co potocznie nazywamy Uniksem. 22 grudnia 1992 roku prawa do Uniksa kupiła od AT&T firma Novell. Nie udało się jej połączyć bardzo udanego projektu swoich usług sieciowych i Uniksa. W 1993 roku ukazała się ostatnia wersja Systemu V, znana jako SVR4.2MP. Novell przekazał prawa do przydzielania marki Unix oraz regulowania Single Unix Specification (SUS)<sup>4</sup> organizacji X/Open<sup>5</sup>. W 1995 roku sprzedaje kod systemu firmie Santa Cruz Operations, a dawne Bell Laboratories firmie Hewlett-Packard. W 1996 roku z połączenia Open Software Foundation i X/Open powstała The Open Group, która przyznaje prawo do posługiwania się nazwą Unix w odniesieniu do systemów komputerowych. W 1997 roku ukazała się druga wersja dokumentu SUS, a pierwsza dostępne publicznie, która wprowadziła wersję 64 bitową oraz wielowątkowość. Kolejna wersja integrująca POSIX z rozszerzeniami przemysłowymi ujrzała światło dzienne w roku 2001.

Prześledźmy historię drugiej gałęzi systemu, która pojawiła się jako konkurencyjna do Systemu V, a mianowicie BSD<sup>6</sup>. Pod koniec lat 70-tych XX wieku, nowy system dotarł również na Uniwersytet Kalifornijski w Berkeley, gdzie w Computer Systems Research Group (CSRG)<sup>7</sup> dodano do niego wiele interesujących udogodnień i zaimplementowano jego nową dystrybucję. Wersja 1 systemu BSD pojawiła się w roku 1977, Wersja 2 już rok później. W roku 1979 ukończono prace nad systemem operacyjnym VAX/BSD, przeznaczonym dla komputerów VAX, który

<sup>1</sup>FSF to podstawowa instytucja sponsorująca projekt GNU. Założona przez Richarda Stallmana z MIT w roku 1985. Jej misją jest tworzenie, ochrona i promocja wolności użytkownika, kopiowania, modyfikowania i rozprowadzania programów komputerowych oraz obrona praw użytkowników *Wolnego Oprogramowania*. Wspiera wolność wypowiedzi, publikacji i zrzeszania się w Internecie, prawo do używania wolnego oprogramowania, szyfrowania prywatnej korespondencji i połączeń sieciowych.

<sup>2</sup>OSF to organizacja założona w 1988 roku w celu utworzenia otwartego standardu systemu operacyjnego Unix. W roku 1996 połączyła się z konsorcjum X/Open Company tworząc The Open Group.

<sup>3</sup>Portable Operating System Interface, co można przetłumaczyć jako przenośny interfejs systemu operacyjnego. Jego powstanie wiąże się z próbą standaryzacji różnych odmian systemu Unix. Pracę nad POSIXem rozpoczęto w roku 1985, a kierowało nimi stowarzyszenie IEEE. Stąd znany jest jako IEEE 1003. Standaryzuje: interfejs programistyczny (API), interfejs użytkownika czyli polecenia systemowe oraz właściwości powłoki.

<sup>4</sup>SUS to dokument opublikowany przez The Open Group. Definiuje on ogólnie jak powinien zachowywać się system operacyjny klasy Unix. Obecnie obowiązuje wersja numer 3 dokumentu. Jest ona w dużej mierze oparta na specyfikacji POSIX.

<sup>5</sup>X/Open to konsorcjum, które realizując idee systemów otwartych czyniło starania w zakresie standaryzacji stosowanych w systemach uniksowych rozwiązań.

<sup>6</sup>Skrót BSD pochodzi od pierwszych liter nazwy Berkeley Software Distribution – instytucji, która miała promować i dystrybuować tę wersję systemu.

<sup>7</sup>CSRG to informatyczna placówka badawczo-rozwojowa Uniwersytetu Kalifornijskiego w Berkeley. Działała do roku 1990. W 1991 roku jej pracownicy założyli firmę Berkeley Software Design Inc., która zajmowała się sprzedażą licencji systemów BSD oraz wsparciem dla niego.



oprócz oryginalnych dodatków BSD posiadał zarządzanie pamięcią wirtualną w postaci występującej we współczesnych systemach. W tym czasie projekt BSD wzbudził zainteresowanie agencji DARPA<sup>8</sup>, a w 1980 otrzymał dofinansowanie na dodanie wymaganych przez DARPA rozszerzeń. Dalszy rozwój zaowocował wersjami 4 i 4.1BSD. Agencja była usatysfakcjonowana i sfinansowała dalsze prace, co zaowocowało dodaniem do dystrybucji nowego systemu plików (Berkeley Fast File System, FFS), obsługi protokołu TCP/IP oraz IPC. W roku 1983 ukazała się wersja 4.2BSD. Na podstawie jej kodu AT&T włączyła obsługę sieci i pamięci wirtualnej do Systemu V. Wersja 4.3 pojawiła się w roku 1986. Dostępny od 1993 roku system 4.4BSD oraz powstające na jego bazie komercyjne i wolne systemy operacyjne (386BSD, NetBSD, FreeBSD) stały się ofiarą procesu o nieprawne korzystanie z kodu zastrzeżonego przez AT&T (potem po odsprzedaniu go przez firmę Novell). W wyniku ugody pomiędzy Uniwersytetem Kalifornijskim w Berkeley a firmą Novell, do której doszło w czerwcu 1994 roku, powstała wersja 4.4BSD Lite, jako pozbawiona spornego kodu wersja 4.4BSD. Została ona uznana za wolną od wszelkich roszczeń. Proces zahamował na długo rozwój rodziny systemów BSD oraz związanych z nią projektów. Niektóre, jak np. 386BSD upadły. Potem ukazała się jeszcze tylko jedna wersja 4.4BSD Lite2, której kod stał się podstawą dla nowszych wersji systemów FreeBSD, NetBSD, OpenBSD, BSD/OS<sup>9</sup> oraz nigdy nie ukończonego Rhapsody firmy Apple Computer.

Można zatem powiedzieć, że system Unix ma jakby dwie gałęzie: System V oraz BSD. Różnią się one w działaniu i konfiguracji wielu podsystemów. Różnice zauważył także przeciętny użytkownik, chociażby w opcjach i działaniu wielu komend. Nazwa Unix przez długi okres powstawania systemu była zastrzeżoną nazwą handlową. Dlatego nie można jej było używać do nazwania swojego systemu. Stąd producenci sprzętu komputerowego, dostarczają własne implementacje systemu Unix pod charakterystycznymi nazwami, jak np. AIX (firmy IBM), HP-UX (Hewlett-Packard) i wiele innych. Żadna z tych implementacji nie jest czystą wersją Systemu V lub BSD. Każda bazuje na jednej z nich, zawierając pewne fragmenty drugiej i oczywiście indywidualne specyficzne rozwiązania.

Należy jeszcze wspomnieć o trzeciej, wielkiej i bardzo popularnej rodzinie systemów uniksowych. Jej początki to rok 1991. Wówczas Linus Torvalds rozpoczął prace nad jądrem systemu operacyjnego o nazwie Linux, które w połączeniu z narzędziami GNU<sup>10</sup> stworzyło funkcjonalnie pełny, uniksopodobny system operacyjny. Na popularność tego systemu operacyjnegołożyły się głównie dwa czynniki. Po pierwsze zaproszenie do jego tworzenia szerokiej społeczności użytkowników (idea systemów otwartych) oraz długa lista architektur sprzętowych, w tym niszowych, dla których jest implementowany. Obecnie wielu producentów systemów uniksowych wprowadza do swojej oferty własne odmiany Linuksa lub aktywnie uczestniczy w rozwijaniu opartych na nich technologii (IBM, SGI, Sun).

## 1.3 Architektura systemu Unix

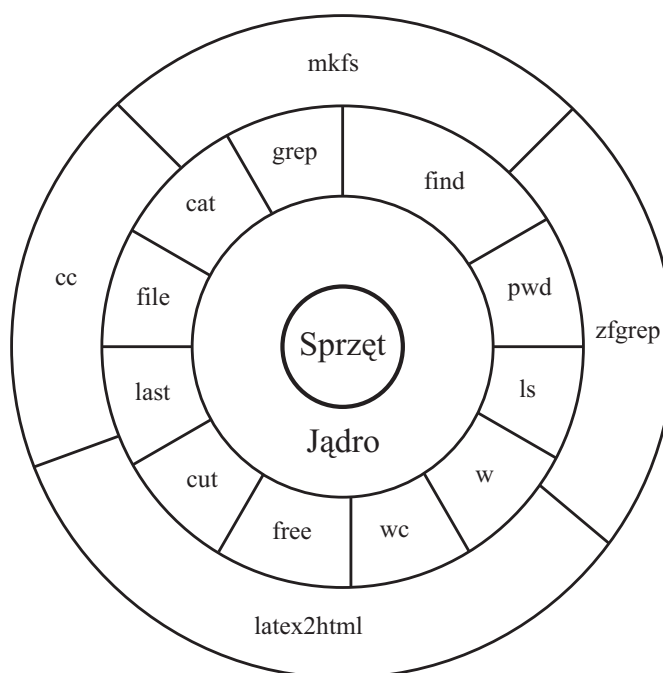
### 1.3.1 Budowa ogólna

Ogólną architekturę systemu operacyjnego Unix przedstawiono na rysunku 1.1. Sprzęt znajdujący się w środku systemu komputerowego dostarcza zasobów biernych, których zarządzaniem zajmuje

<sup>8</sup>DARPA to skrót od Defense Advanced Research Projects Agency, co można przetłumaczyć jako Agencja Zaawansowanych Projektów Badawczych Obrony. Należy do Departamentu Obrony Stanów Zjednoczonych i zajmuje się rozwojem technologii wojskowej. Na jej zamówienie powstała rodzina protokołów sieciowych TCP/IP

<sup>9</sup>Sprzedając licencje oraz wsparciem technicznym dla tej wersji systemu zajmowała się firma BSDI – Berkeley Software Design Inc. W 2001 roku zostało przejęte przez Wind River Systems, a następnie przez Telnet Systems. Wind River zaprzestał dystrybucji BSD/OS w 2003 roku, a w 2004 wycofał wsparcie techniczne.

<sup>10</sup>To rekurencyjny akronim pochodzący od słów GNU'S Not Unix (GNU to nie Unix). To uniksopodobny system operacyjny złożony wyłącznie z wolnego oprogramowania. Był pierwszym projektem FSF.



Rysunek 1.1: Architektura systemu operacyjnego Unix.

się system operacyjny. Bezpośredni dostęp do zasobów sprzętowych ma jedynie jądro systemu operacyjnego. W przybliżeniu można powiedzieć, że jest ono programem zbudowanym z funkcji wyposażonych w interfejsy dostępu do sprzętu. Stąd zmiana konfiguracji sprzętowej, najczęściej pociąga za sobą zmianę konfiguracji jądra polegającą na dodaniu do niego funkcji umiających komunikować się z nowym sprzętem.

W kolejnej warstwie znajdują się komendy (polecenia) systemowe i programy użytkowe. Istotnym jest spostrzeżenie, że nie mają one bezpośredniego dostępu do sprzętu. Konsekwencje takiego podejścia są dwie. Po pierwsze są one niezależne od platformy sprzętowej, a więc łatwo przenaszalne. Po drugie kontrolowany przez bezpieczne i sprawdzone funkcje jądra dostęp do sprzętu powoduje, że proces nie zawłaszczy żadnego zasobu. Oznacza to np., że proces nie jest w stanie zapisywać do pliku, do którego nie ma praw zapisu, czy ogólniej w dowolnym miejscu dysku. Jak zostało to zrealizowane? Proces może pracować w dwóch trybach. W *trybie użytkownika* proces wykonuje instrukcje użytkownika wykorzystując jedynie procesor. Dostęp do jakiegokolwiek urządzenia (np. w celu wykonania operacji wejścia/wyjścia) powoduje przełączenie jego wykonania w *tryb jądra*, który polega na wykonaniu danej operacji przez funkcje jądra systemu. Procesy komunikują się z jądrem systemu za pomocą zdefiniowanego zbioru funkcji systemowych. Funkcje te zlecają jądru wykonanie dla zlecającego procesu różnych operacji i realizują ogólnie rozumianą wymianę danych między jądrem i programem. W Systemie V jest ponad 60 funkcji systemowych, z czego praktycznie wykorzystuje się około połowy.

Inne programy użytkowe mogą korzystać z programów i komend systemowych warstwy niższej. Stąd pojawienie się na rysunku 1.1 najbardziej zewnętrznej warstwy, w której znalazł się jako przykład kompilator `cc`. W wielu implementacjach wywołuje on program preprocesora, kompilator i program konsolidujący (linkujący). Styl programowania polegający na łączeniu istniejących programów w celu realizacji nowego jest właściwy dla systemów uniksowych. Stąd warstw repre-

zentuujących programy użytkowe w praktyce może być więcej.

### 1.3.2 Architektura jądra

Z dobrym przybliżeniem można powiedzieć, że Unix wspiera iluzję *systemu plików*, który posiadał miejsce do przechowywania informacji oraz *procesów*, które stanowią aktywną część systemu komputerowego. Stąd dwie jednostki, plik oraz proces, stanowią podstawowe pojęcia w modelu systemu. Rysunek 1.2 przedstawia ich wzajemne związki. Stanowi on „powiększenie” warstwy jądra z rysunku 1.1 wraz z zaznaczeniem warstw sąsiednich i jest bardziej modelem logicznym, gdyż w rzeczywistych implementacjach systemu działanie niektórych modułów zależy od pracy innych.

Funkcje systemowe oraz interfejs biblioteczny stanowią granicę między programami użytkownika a jądrem systemu. Zadaniem funkcji bibliotecznych jest odwzorowanie wywołań funkcji w programach na wywołania funkcji systemowych, umożliwiającym dostęp do jądra systemu. Funkcje biblioteczne są łączone z kodem programu użytkownika w czasie jego kompilacji. Na rysunku 1.2 zbiór funkcji systemowych podzielono na te, które współpracują z systemem plików i te, które współpracują z podsystemem sterowania procesami. Zadaniem podsystemu plików jest zarządzanie plikami, przydzielanie im pamięci dyskowej, administrowanie pamięcią dyskową, sterowanie dostępem do plików i udostępnianie procesom użytkowników danych zapisanych w plikach zgodnie ze zdefiniowanymi prawami dostępu. Procesy komunikują się z podsystemem plików korzystając z funkcji systemowych, z których podstawowe to: *open*, *close*, *read*, *write*, *stat* (formułowanie zapytań o prawa dostępu do pliku), *chown* oraz *chmod*.

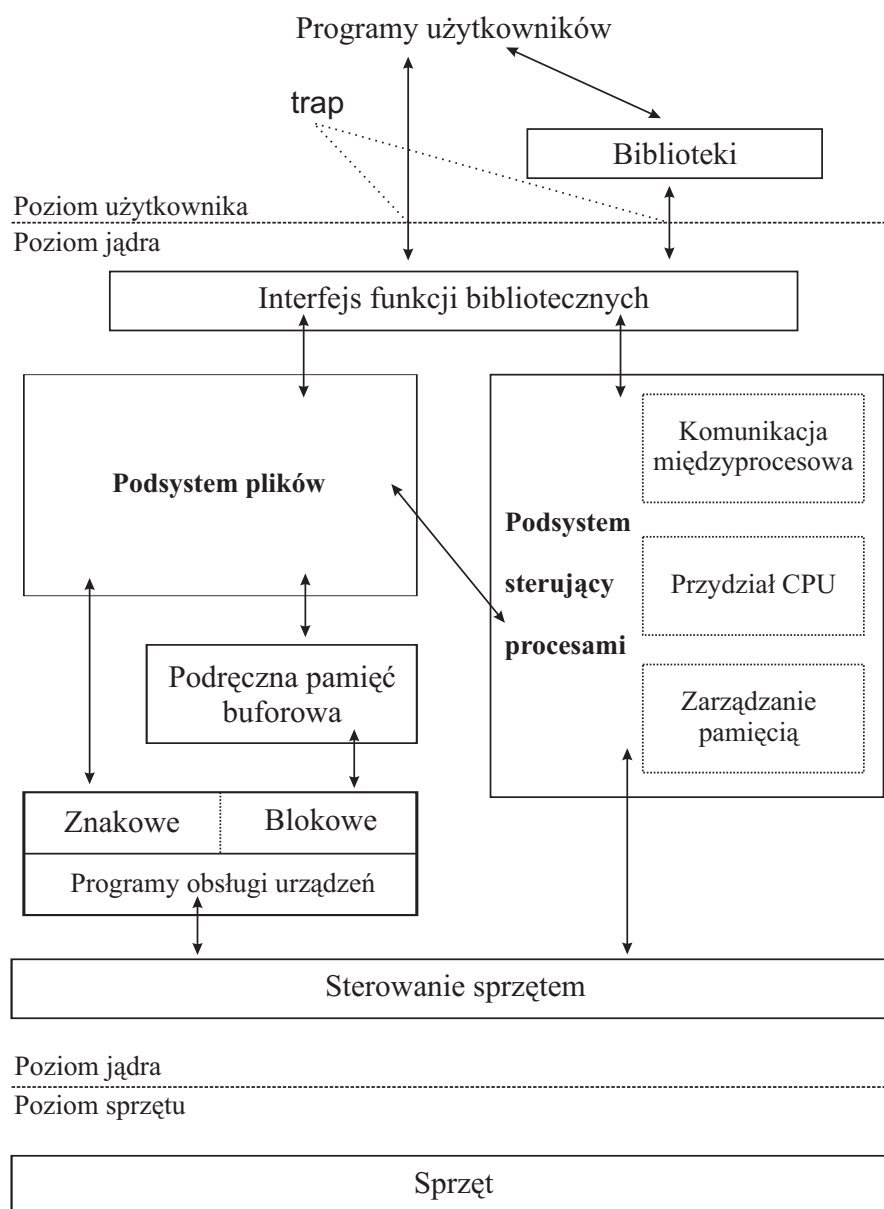
Podsystem plików udostępnia dane w nim przechowywane za pomocą mechanizmu buforowania, który służy regulacji przepływu danych pomiędzy jądrem systemu, a urządzeniami pamięci masowej. Mechanizm ten współpracuje z procedurami obsługi urządzeń blokowych wejścia-wyjścia w celu inicjalizacji transmisji danych do i z jądra systemu. Procedury te w praktyce są modułami jądra, których zadaniem jest sterowanie pracą urządzeń wejścia-wyjścia. Blokowe urządzenia wejścia-wyjścia są urządzeniami o dostępie swobodnym lub też procedury ich obsługi udostępniają je w systemie jako takie urządzenia. Podsystem wejścia-wyjścia współpracuje także z procedurami obsługi znakowych (surowych) urządzeń wejścia-wyjścia nie korzystając w tym przypadku z mechanizmu buforowania.

Zagadnienia związane z podsystemem obsługi plików (inaczej pamięci masowej) stanowią treść rozdziału 3.

Podsystem sterowania procesami jest odpowiedzialny za szeregowanie procesów i zarządzanie pamięcią operacyjną, synchronizację procesów oraz komunikacją między nimi. Współpracuje on z podsystemem plików w momencie uruchamiania nowego procesu w celu załadowania jego segmentów do pamięci operacyjnej oraz podczas przenoszenia fragmentów procesu do przestrzeni wymiany w celu zwolnienia miejsca w pamięci operacyjnej.

Podstawowymi funkcjami służącymi sterowaniu procesami są: *fork* – sklonuj bieżący proces, *exec* – nałóż obraz programu na program sklonowanego procesu, *exit* – zakończ wykonywanie procesu, *wait* – dokonaj synchronizacji zakończenia wykonywania bieżącego procesu z procesem utworzonym przez niego funkcją *fork*, *brk* – skontroluj rozmiar pamięci przydzielonej procesowi oraz *signal* – zdefiniuj indywidualną reakcję procesu na zajście określonego zdarzenia.

Zadaniem modułu zarządzania pamięcią jest sterowanie przydzielaniem pamięci dla procesów. Jeśli w danym momencie czasu nie ma dość pamięci fizycznej dla wszystkich procesów, to w zależności od stopnia deficytu, jądro przesyła ich fragmenty lub całe procesy między obszarem wymiany, a pamięcią operacyjną tak, aby każdy z nich miał szansę się wykonać. Proces realizujący te zadania będziemy nazywać *procesem wymiany*. Można spotkać się z nazwą *scheduler* dla zaznaczenia faktu, iż szereguje on żądania przydziału pamięci i współpracuje z procesem



Rysunek 1.2: Schemat budowy jądra systemu operacyjnego Unix.

przydziału CPU nazywanym *CPU scheduler*.

Moduł przydziału CPU realizuje politykę przydzielania go procesom. Ustala on kolejność wykonania procesów oraz przydziela im procesor do chwili ich zakończenia lub wyczerpania kwantu czasu, na który procesor był im przydzielony. W takiej sytuacji proces szeregujący wybiera z kolejki procesów gotowych do wykonania ten o największym priorytecie i jemu przydziela procesor.

Moduł komunikacji międzyprocesowej realizuje wymianę informacji między procesami na kilka sposobów. Dostępne mechanizmy umożliwiają asynchroniczne sygnalizowanie pewnych zdarzeń

oraz synchroniczną transmisję komunikatów pomiędzy procesami.

Zagadnienia związane z podsystemem sterowania procesami zostały omówione w rozdziale 4.

Moduł sterowania sprzętem jest odpowiedzialny za obsługę przerwania oraz komunikowanie się z urządzeniami systemu komputerowego. Źródłem przerwania może być np. dysk informujący w ten sposób o zakończeniu wykonywania operacji wejścia-wyjścia polegającej na odczycie danych z pliku przez aktualnie wykonywany proces. Pojawienie się przerwania skutkuje przerwaniem pracy procesora podczas wykonywania instrukcji procesu. Jądro systemu operacyjnego może wznowić wykonywanie przerwanej operacji po zakończeniu obsługi przerwania. Przerwania są obsługiwane przez funkcje stanowiące kod jądra, a wywoływane w kontekście wykonywanego właśnie procesu.

## 1.4 Zalety i wady systemu

Źródło popularności systemu Unix należy upatrywać w następujących jego zaletach:

- Większość kodu systemu została napisana w języku wysokiego poziomu, dzięki czemu jest on wydajny, łatwo modyfikowalny oraz przenaszalny między różnymi platformami sprzętowymi.
- Jest wyposażony w prosty, intuicyjny interfejs.
- Umożliwia budowanie oprogramowania w sposób hierarchiczny, tzn. tworzenie nowych programów z wykorzystaniem już istniejących.
- Posiada hierarchiczny system plików, który umożliwia łatwe zarządzanie plikami oraz efektywną implementację programów na nim operujących.
- Używa spójnego formatu plików, traktując każdy jako strumień bajtów. Ułatwia to operacje na ich zawartości.
- Zapewnia prosty i spójny interfejs z urządzeniami systemu komputerowego. Każde z nich jest reprezentowane w systemie operacyjnym przez plik służący do jego zarządzania.
- Zaskłania przed użytkownikiem architekturę sprzętową udostępniając jedynie interfejs w postaci funkcji bibliotecznych, co zapewnia przenaszalność programów.
- Jest systemem wieloużytkownikowym i wielozadaniowym pracującym w trybie z wywłaszczaniem.

Zasadniczymi wadami systemu operacyjnego Unix są:

- Duża liczba plików konfiguracyjnych o różnym formacie, często nazwie, występujących w różnych dystrybucjach systemu.
- Pomimo wysiłków wielu organizacji niski poziom unifikacji różnych dystrybucji, który dla zwykłego użytkownika objawia się różnymi opcjami w podstawowych komendach.

## 1.5 Konwencje przyjęte w kolejnych rozdziałach

Zawirowania spowodowane komercjalizacją kodu Uniksa doprowadziły do powstania wielu dystrybucji systemu. Jak zaznaczono wcześniej system posiada dwie gałęzie, a mianowicie pochodzący z AT&T System V oraz BSD. W praktyce mamy do czynienia z implementacjami różnych

producentów, którzy w swych produktach wykorzystują rozwiązania obu rodzin. Ich ilość można szacować na kilkadziesiąt. Pomimo wielu wysiłków nie udało się ich ujednolicić. Dochodzi jeszcze bardzo liczna rodzina uniksopodobnych systemów linuksowych. Szacuje się, że obecnie na rynku jest obecnie dostępnych ok. 700 różnych dystrybucji. Rodzi się zatem pytanie jak napisać podręcznik, który opisuje system operacyjny występujący w tak wielu dystrybucjach, zadowalając przy tym miłośników każdej z nich? Ze względów technicznych jest to nie możliwe. Stąd powstał pomysł na realizację kilku tomów. W tomach pierwszym i drugim zajmiemy się podstawami, a zagadnienia konfiguracji zostaną omówione tak, by działały na większości najpopularniejszych dystrybucji. Pojawia się również rozwiązania specyficzne, jak np. *Logical Volume Manager* (LVM) występujący w systemie AIX czy Fedora (omówiony na jej przykładzie) lub *Vinum* z rodziny BSD (omówiony na przykładzie FreeBSD). Należy je traktować jako przykłady rozwiązań pewnych podsystemów, które kiedyś pomogą w wyborze dystrybucji najlepiej spełniającej nasze wymagania. W tomie trzecim i kolejnych zostaną przedstawione przykłady konkretnych konfiguracji na wybranych dystrybucjach. Stanowią one rozwiązania zagadnień, które najczęściej pojawiają się podczas realizacji nowoczesnych systemów informatycznych. Ciągły rozwój systemów uniksowych powoduje, że tom trzeci i kolejne będą miały charakter stale niedokończonych, aczkolwiek mam nadzieję, że będą pojawiały się ich kolejne wydania.

## Rozdział 2

# Użytkownicy i grupy użytkowników w systemie Unix

Zgodnie z definicją, system operacyjny służy do udostępniania zasobów sprzętowych procesom, czyli uruchomionym programom. Dla naszych potrzeb procesy możemy podzielić na systemowe, czyli pełniące w systemie określone zadania wynikające z jego funkcji oraz uruchomione przez użytkowników. Oczywiście jest, że zwłaszcza te drugie będą dążyły do maksymalnego wykorzystania dostępnych zasobów w systemie. Pojawia się zatem problem kontroli dostępu do zasobów, sposobu i dążenie do efektywnego ich wykorzystywania, a w systemach wieloużytkownikowych ochrony zasobów będących własnością jednego użytkownika przed działaniami innych użytkowników. Stąd podstawy efektywnego i bezpiecznego funkcjonowania systemu komputerowego tkwią już w poprawnym zarządzaniu użytkownikami.

Użytkownik w systemie operacyjnym jest zasobem abstrakcyjnym. W praktyce oznacza to tyle, że jądro systemu rozpoznaje go po numerze identyfikacyjnym (tzw. User Identification Number, UID). Na tej podstawie ustalany jest właściciel danego zasobu (procesu lub pliku) oraz dopuszczane lub nie operacje wykonywane na tym zasobie. Stwierdzenie, czy użytkownik ma prawo do danej operacji polega w praktyce na porównaniu identyfikatora właściciela procesu (najczęściej jest to identyfikator użytkownika, który dany proces uruchomił) z identyfikatorem właściciela zasobu, na którym proces ma operować. Wiemy, że porównanie dwóch liczb całkowitych, w przeciwieństwie do porównywania napisów, jest operacją wykonywaną przez jednostki arytmetyczno-logiczne procesora bardzo szybko. Stąd pomysł na „oznaczanie” użytkowników numerami. Niestety, liczby pamiętamy gorzej niż pseudonimy, kalambury czy skróty nazw nadane często na zasadzie skojarzenia. Dodatkowo, przypisanie człowiekowi numeru może zostać różnie odebrane.

## 2.1 Podstawowe pliki konfiguracyjne w dystrybucji Fedora

### 2.1.1 Plik */etc/passwd*

Pogodzenie sprzecznych wymagań polega na używaniu nazw przez użytkowników, zaś numeru przez system operacyjny. Należy jeszcze zaproponować sposób przejścia między jednym, a drugim opisem oraz format przechowywania tej informacji. W tym celu powstał plik o nazwie *passwd*, znajdujący się w katalogu */etc*. Zawartość pliku może zostać odczytana przez każdego użytkownika w systemie. Modyfikacja zawartości przez zwykłego użytkownika jest możliwa tylko poprzez

wykonanie komend o specyficznych prawach dostępu<sup>1</sup>. Fragment pliku przedstawiono poniżej.

```

1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

```

Jest to plik tekstowy o budowie linijkowej. Jedna linijka opisuje jednego użytkownika. Każda linijka składa się z 7-dmiu pól oddzielonych znakiem specjalnym, jakim jest ":"(dwukropek). Pola zawierają:

1. Nazwę użytkownika w systemie. Musi ona być niepowtarzalna, a zatem nie powinniśmy mieć więcej niż jednego użytkownika w systemie o takiej samej nazwie.
2. Drugie pole ma znaczenie historyczne. Dawniej znajdowała się w nim postać zakodowana hasła<sup>2</sup>. Jednak wzrost mocy obliczeniowej procesorów spowodował, że rozwiązanie to przestało być bezpieczne. Wynika to z faktu, iż prosty program realizujący algorytm brutalnego łamania hasła, a więc generowania wszystkich ciągów z zadanego zbioru znaków, a następnie kodujący je znaną funkcją<sup>3</sup> i porównujący, czy otrzymana postać zakodowana jest identyczna z odczytaną uruchomiony na przeciętnym procesorze może znaleźć hasło w ciągu kilkunastu do kilkudziesięciu godzin. Stąd postanowiono przenieść postać zakodowaną hasła w „bardziej bezpieczne miejsce”. Obecnie pole to jest wykorzystywane przez niektóre dystrybucje do przechowywania informacji o stanie hasła. Rozwiązanie to jest jednak mocno indywidualne. Przykładowo, system AIX . Gdzie przechowywana jest postać zakodowana hasła? Rozwiązań jest kilka. Pierwsze to tzw. *shadow* (przesłanianie hasła). Polega ono na zapisaniu postaci zakodowanej hasła w pliku, którego zawartość może zostać odczytana tylko przez administratora systemu. Plik ten nazywa się *shadow* i znajduje się w katalogu */etc*. Drugie rozwiązanie idzie o krok dalej. Tworzy w katalogu */etc* katalog (w AIX o nazwie *security*) do którego przejść może jedynie administrator, a w nim plik zawierający m. in. postać zakodowaną hasła. Zawartość pliku może zostać odczytana jedynie przez administratora systemu. W dalszej części zajmiemy się pierwszym rozwiązaniem, jako bardziej elastycznym. Jeszcze inne podejście zastosowano w rodzinie BSD. Tam plik */etc/passwd* został zachowany dla „zgodności” z innymi dystrybucjami. Hasło w postaci zakodowanej jest przechowywane w pliku */etc/master.passwd*, którego zawartość może zostać odczytana jedynie przez użytkownika *root*. Dodatkowo, dla podniesienia wydajności, informacje o użytkownikach przechowywane są w bazie danych, której plikiem jest plik */etc/pwd.db*. Zawartość tego pliku może zostać odczytana przez każdego użytkownika w systemie. Nie stanowi to luki w bezpieczeństwie, gdyż informacja zapisana w bazie danych odpowiada zawartości pliku */etc/passwd*, a ta jest dostępna dla każdego użytkownika systemu.
3. Trzecie pole przechowuje numer identyfikacyjny użytkownika (UID). Powinien być on niepowtarzalny, gdyż służy on do rozpoznawania użytkownika w systemie. Można jednak przypisać ten sam numer identyfikacyjny różnym użytkownikom, jeśli ich konta są wykorzystywane przez tą samą osobę. Mówimy wówczas, że osoba ta może podłączać się do systemu

<sup>1</sup>Zagadnienie to zostało dokładnie omówione w następnym rozdziale

<sup>2</sup>Hasło jest przechowywane w systemie tylko w postaci zakodowanej lub zaszyfrowanej, zależnie od użytej funkcji. Postać jawna, czyli wykorzystywana przez użytkownika nie jest nigdzie dostępna.

<sup>3</sup>Najczęściej jest to funkcja *crypt*, której opis można znaleźć w 3 rozdziale manuala, komendą *man 3 crypt*.



w różnych kontekstach. Poszczególni użytkownicy systemu mogą mieć przykładowo inaczej zdefiniowane środowisko pracy, ale identyczną listę plików będących ich własnością. W zdecydowanej większości systemów, użytkownik o najszerszych uprawnieniach ma nazwę *root* i numer identyfikacyjny 0. Kolejne numery przypisane zostały użytkownikom funkcyjnym. Nie są oni wykorzystywani do podłączania się do systemu, lecz służą głównie do przekazywania praw własności do ściśle określonych plików zawierających komendy służące do administrowania jakimś fragmentem systemu komputerowego (np. kolejkami zadań do drukowania i drukarkami). Pozwala to na przekazanie części czynności administracyjnych innym użytkownikom i odciążenie, zazwyczaj jednego, administratora. Zwykłym użytkownikom przypisuje się zazwyczaj numery od pewnej wartości w górę (w jednych systemach od 200, w innych od 500 czy 1000).

4. W czwartym polu znajduje się numer identyfikacyjny grupy podstawowej użytkownika. Podział użytkowników na grupy wprowadzono m. in. w celu ułatwienia zarządzania nimi. Każdy użytkownik może należeć do wielu grup, ale jedna jest jego grupą podstawową. Jest ona decydująca w kwestiach praw własności. Przykładowo plik utworzony przez danego użytkownika, jako właściciela indywidualnego będzie miał właśnie jego, a właścicielem grupowym zostanie grupa podstawowa tegoż użytkownika. Plik zawierający podstawową definicję grup to */etc/group*.
5. Pole piąte to opis użytkownika. Od administratora systemu zależy, co w tym polu się znajdzie. W szczególności może pozostać puste. Najczęściej wpisywane jest nazwisko i imię użytkownika (w dowolnej kolejności). Niektóre komendy (np. *chfn*, *finger*) porządkują to pole dzieląc je na 4 części oddzielone przecinkami. Kolejne części są interpretowane jako: nazwisko i imię, numer pokoju, numer telefonu służbowego i numer telefonu domowego.
6. W szóstym polu znajduje się bezwzględna ścieżka dostępu do katalogu domowego użytkownika. Po pomyślnym podłączeniu się do systemu katalog ten stanie się katalogiem bieżącym. Jeśli natomiast z jakiegoś powodu katalog ten jest niedostępny, bieżącym katalogiem będzie katalog główny. Można spotkać się również z konfiguracją, w której brak dostępu do katalogu domowego uniemożliwia podłączenie się do systemu.
7. Ostatnie, siódme, pole zawiera bezwzględną ścieżkę dostępu do podstawowego interpretera poleceń użytkownika. Jeśli ścieżka ta z jakiś powodów jest błędna, to w większości systemów zostanie uruchomiony interpreter domniemany. W innych użytkownik nie będzie mógł podłączyć się do systemu. Lista dostępnych programów, które mogą pełnić funkcje interpreterów poleceń jest najczęściej przechowywana w pliku */etc/shells*.

### 2.1.2 Plik */etc/shadow*

Zobaczmy jak wygląda zawartość pliku */etc/shadow*, w którym przechowywane są m.in. hasła użytkowników. Jego fragment, pochodzący z dystrybucji Fedora 9 przedstawiono poniżej.

```

1 root:$6$i1PbKcizy5vjYEA5$SealtcoZREoWG7K/R3DFF84/gbVbmU50mpWHN7Qv5RB5h1HozT6
2      kwGTZzLqyhjExgNYkhK2QYX8PhcgQe0rw7/:14214:0:99999:7:::
3 bin:*:14214:0:99999:7:::
4 daemon:*:14214:0:99999:7:::
5 .....
6 wacek:$6$gZwWgJK5$g5zQNJk3Lh8LqhvzzSZ5pYR.oDrHwR5Lr3bLCK2d6wxhn/tJPQqzV/10CENa
7      gIY07JEa8cUcqzuZvyqB4S82e0:14214:0:99999:7:::
8
```

```

9  maciek!!!:14270:0:1:7:1::
10 jan::14270:0:1:7:::

```

Jak widać jest to plik tekstowy o budowie linikowej. Każda linijka zawiera definicję jednego użytkownika i składa się z 9-ciu pól oddzielonych znakiem ":". Pola mają następujące znaczenie:

1. Pierwsze pole przechowuje nazwę użytkownika w systemie.
2. W polu drugim przechowywane jest hasło w postaci zaszyfrowanej. Jeszcze do niedawna, do szyfrowania haseł wykorzystywano najczęściej jednokierunkową funkcję skrótu MD5. Postać zaszyfrowana była generowana z postaci jawnej i tzw. soli, czyli ciągu znaków dodawanego do ciągu szyfrowanego. Najczęściej jest to ciąg znaków przypadkowych. Jego unikalność zagwarantuje teoretycznie, że nie pojawią się w systemie dwa hasła o identycznej postaci zaszyfrowanej, nawet jeśli dwóch użytkowników wprowadzi takie samo hasło. Jednak tylko teoretycznie, gdyż już parę lat temu wskazano na występowanie w tej funkcji tzw. kolizji [8]. Polegają one na generowaniu takiego samego skrótu dla różnych ciągów znaków wejściowych. Zastosowanie algorytmu detekcji kolizji pozwala na 17-to krotne skrócenie czasu łamania hasła metodą brutalną. Stąd zaczęto poszukiwać nowych rozwiązań i obecnie systemy stosują powszechnie algorytmy SHA-256 lub SHA-512.

Hasła przechowywane są w następującej postaci ogólnej, *\$identyfikator\$sól\$hasło*, gdzie *identyfikator* oznacza wykorzystaną funkcję kodującą, *sól* dodany do postaci jawnej ciąg znaków, *hasło* postać zakodowaną hasła. Krótką charakterystykę wykorzystywanych funkcji przedstawiono w tabeli 2:

Identyfikator	Funkcja	Długość postaci zakodowanej
1	MD5	22
2a	Blowfish	
md5	Sun MD5	22
5	SHA-256	43
6	SHA-512	86

Tabela 2.1: Podstawowe metody kodowania haseł stosowane w systemach uniksowych.

Ale w przedstawionym fragmencie, w drugim polu znajdują się jeszcze trzy inne możliwości, które na pewno nie są postaciami zaszyfrowanymi hasła. Najniebezpieczniejszą jest pole puste. Oznacza ono w praktyce, że ustawione zostało hasło puste. W systemach wieloużytkownikowych sytuacja taka nie powinna mieć miejsca. Kolejny przypadek to \*. Oznacza ona, że nie możemy podłączać się do systemu jako użytkownik, którego opisuje ta linia. Przedostatnia możliwość, !! oznaczają, że żadne hasło nie pasuje, ale jeśli zostanie ustawione, to użytkownik będzie mógł się do systemu podłączać. W polu tym może pojawić się również postać zaszyfrowana hasła, poprzedzona znakiem !. Oznacza to, że hasło zostało zablokowane. Usunięcie znaku ! spowoduje jego odblokowanie.

3. W trzecim polu przechowywana jest informacja o liczbie dni, które upłynęły od 1 stycznia 1970 roku do chwili ostatniej zmiany hasła. W oparciu o nią można prowadzić politykę odpowiedniej częstotliwości zmiany hasła. Wartość 0 wymusza na użytkowniku zmianę hasła przy następnym podłączaniu się do systemu.
4. Pole czwarte przechowuje minimalny wiek życia hasła w dniach. W praktyce oznacza on liczbę dni liczoną od momentu ostatniej zmiany hasła, przed upływem których hasło nie

może być zmienione. Wartość 0 oznacza, że może ono zostać zmienione w każdej chwili. Minimalny wiek życia hasła pozwala na efektywne prowadzenie polityki bezpieczeństwa z listą historii haseł.

5. Piąte pole zawiera maksymalny wiek życia hasła, a więc liczbę dni po upływie których licząc od momentu ostatniej zmiany hasła, hasło będzie musiało być zmienione. Ustawiona wartość 99999 powoduje, że za ok. 234 lata hasło straci ważność.
6. Aby konieczność zmiany hasła nie była zaskoczeniem, w polu szóstym zapisana została liczba dni dzielących danego użytkownika od momentu, w którym hasło będzie musiało być zmienione. Przez ten czas, w trakcie podłączania się do systemu, będzie pojawiał się stosowny komunikat.
7. W polu siódmym przechowywana jest liczba dni, po upływie których licząc od momentu upłynięcia maksymalnego wieku życia (wygaśnięcia ważności) hasła, konto zostanie zablokowane. Krótki komentarz. Jeśli użytkownik nie podłącza się do systemu, to nie ma możliwości wymuszenia na nim odpowiedniej częstości zmiany hasła. Ustawienie maksymalnego wieku życia hasła po prostu nie działa. Jeśli hasło pozostaje nie zmienione, to rośnie prawdopodobieństwo powodzenia włamania do systemu przez takie konto. Stąd automatyczne blokowanie kont użytkowników, którzy dawno nie podłączali się do systemu. Brak jakiegokolwiek wartości w tym polu oznacza, że ograniczenie to nie działa.
8. Pole ósme zawiera liczbę dni po upływie których licząc od 1 stycznia 1970 roku konto zostanie automatycznie zablokowane. Ustawienie to stosuje się dla kont zakładanych okazjnie, np. dla studentów odbywających praktykę wakacyjną. Po jej zakończeniu nie musimy pamiętać, aby blokować konto. Podobnie, jak w polu poprzednim, brak jakiegokolwiek wartości oznacza, że ograniczenie to nie działa.
9. Pole dziewiąte jest rezerwowym. System w podstawowej konfiguracji ignoruje je. Może być wykorzystywane przez niektóre aplikacje.

Jak widać technika *shadow* dostarcza podstawowych mechanizmów do prowadzenia polityki bezpieczeństwa odnośnie haseł użytkowników. Możliwe staje się ograniczanie częstości zmian hasła, czyli minimalnego wieku hasła oraz maksymalnego wieku hasła. Mamy mechanizm blokowania konta, jeśli użytkownik nie podłączał się do systemu, a wiek hasła przekroczył okres ważności. Możliwe jest także automatyczne blokowanie konta, które z założenia miało być dostępne przez określony okres czasu. Zauważmy, że możliwości te nie są funkcjonalnie pełne. Ograniczenie możliwości zbyt częstej zmiany hasła wydaje się być sprzeczne z zasadami bezpieczeństwa, gdyż im krócej hasło obowiązuje, tym mniejsze prawdopodobieństwo jego złamania. Jeśli chodzi o ograniczenie maksymalnego wieku hasła, to bez dodatkowych ograniczeń, po upływie jego ważności może ono zostać zmienione na to samo. Możliwość uzupełnienia mechanizmów bezpieczeństwa dostarcza system PAM, który został omówiony w podrozdziale 2.9.

### 2.1.3 Plik */etc/group*

Grupy użytkowników zostały wprowadzone dla ułatwienia zarządzania użytkownikami, w celu łatwego przydzielenia wybranych czynności administracyjnych zwykłym użytkownikom, a w konsekwencji podniesieniu poziomu bezpieczeństwa systemu. W najprostszy sposób można to zrealizować wprowadzając pojęcie właściciela grupowego. Jest nim grupa użytkowników, co w praktyce oznacza, że każdy jej członek ma takie właśnie prawa do zarządzania obiektem. Jeśli przykładowo właścicielem grupowym pliku */sbin/shutdown* jest grupa *root*, to każdy użytkownik, który

do niej należy może dokonywać zamknięcia lub restartu systemu<sup>4</sup>. Podział użytkowników na grupy ma przede wszystkim zastosowanie przy współdzieleniu plików. Ma to miejsce w przypadku realizacji projektu przez kilku użytkowników. W systemie tworzy się wówczas grupę, do której przypisuje się uczestników projektu. Mają oni możliwość współdzielenia plików, których właścicielem grupowym jest właśnie ta grupa, zgodnie z przypisanymi jej prawami dostępu.

Z krótkiego wprowadzenia można wysnuć dwa wnioski. Po pierwsze, wprowadzenie grup użytkowników, które mają prawa własności, ułatwia współdzielenie plików. Jednak konieczność definiowania grupy w celu umożliwienia współdzielenia dostępu jest kłopotliwe i prowadzi do powstania dużej liczby grup użytkowników w systemie. Utrudnia to administrowanie. Po drugie omyłkowe dodanie niewłaściwego użytkownika np. do grupy posiadającej uprawnienia do uruchamiania programów służących do zarządzania systemem może mieć negatywne konsekwencje dla całej społeczności użytkowników.

Przejdźmy do omówienia budowy podstawowego pliku opisującego grupy użytkowników zdefiniowane w systemie, a mianowicie znajdującego się w katalogu */etc* pliku *group*. Jego fragment przedstawiono poniżej.

```

1 root:x:0:root
2 bin:x:1:root,bin,daemon
3 daemon:x:2:root,bin,daemon
4 .....
5 wacek:x:500:
6 maciek:x:501:

```

Zawartość tego pliku może zostać odczytana przez każdego użytkownika w systemie, modyfikowana jednak tylko przez użytkownika *root*. Jest to plik tekstowy o budowie wierszowej. Jeden wiersz opisuje jedną grupę zdefiniowaną w systemie i składa się z 4 pól oddzielonych znakiem ":". Kolejne pola oznaczają:

1. W pierwszym polu znajduje się nazwa grupy w systemie. Podobnie, jak w przypadku nazwy użytkownika, powinna ona być niepowtarzalna w systemie.
2. Drugie pole ma znaczenie historyczne. Dawniej przechowywało ono zakodowaną postać hasła grupowego. Obecnie, ze względów bezpieczeństwa zostało ono przeniesione w inne miejsce, zależne od przyjętego w dystrybucji rozwiązania. W najczęściej stosowanym zasłanianiu plików z hasłami (*shadow*) znajduje się ono w pliku */etc/gshadow*, który odczytać lub modyfikować programi administracyjnymi może jedynie administrator systemu. Niektóre dystrybucje Poszły o krok dalej i podobnie jak w przypadku haseł użytkowników, hasła grupowe przechowywane są w pliku, który odczytać lub modyfikować może jedynie administrator systemu, a plik ten znajduje się w katalogu, do którego wejść może również tylko administrator.
3. Pole trzecie przechowuje numer identyfikacyjny grupy w systemie (GID). Numer ten powinien być niepowtarzalny w obrębie systemu, czyli nie powinny znaleźć się w nim dwie grupy o takim samym numerze. Wynika to z konieczności zapewnienia jednoznacznego ustalenia właściciela grupowego pliku lub procesu celem określenia możliwości wykonania na nich danej operacji. Zdarzają się jednak przypadki, gdy kilka grup ma ten sam numer identyfikacyjny. Ma miejsce wówczas, gdy konieczne jest aby wybrane obiekty w systemie miały kilku właścicieli grupowych. W większości systemów grupy predefiniowane, a więc dostępne

<sup>4</sup>W praktyce uruchomiony proces sprawdza jaki jest efektywny numer identyfikacyjny użytkownika, który ten proces uruchomił. Jeśli jest on różny od 0, to proces dalej się nie wykonuje.

w systemie po jego instalacji, mają numery zaczynające się od 0. Grupy dodawane w trakcie pracy systemu otrzymują numery od pewnej wartości. Zależy ona od ustawień zapisanych w plikach konfiguracyjnych danej dystrybucji systemu i wynosi przykładowo 200, 500 lub 1000.

4. W czwartym polu znajdują się nazwy użytkowników systemu oddzielone przecinkiem, dla których grupa ta jest grupą dodatkową. Puste pole oznacza, że nie jest ona dodatkową dla żadnego użytkownika.

#### 2.1.4 Plik */etc/gshadow*

Fragment pliku */etc/gshadow* pochodzący z dystrybucji Fedora 9 przedstawiono poniżej.

```
1 root:::root
2 bin:::root,bin,daemon
3 daemon:::root,bin,daemon
4 sys:::root,bin,adm
5 adm:::root,adm,daemon
6 .....
7 wacek:$1$WW.9ucZL$aal/x61r/80N/ZyapZ9Ej.:root:
8 maciek:::
```

Jest to plik tekstowy o budowie linijkowej. Każda linia opisuje jedną grupę i składa się z czterech pól. Kolejne pola zawierają:

1. Pierwsze nazwę grupy.
2. W drugim polu przechowywana jest postać zakodowana hasła. Jego format jest dokładnie taki sam, jak w drugim polu linii w pliku */etc/shadow*. Można z niej odczytać identyfikator algorytmu szyfrującego oraz sól. Symbole "\*" oraz "!" oznaczają odpowiednio, że wykorzystanie hasła grupowego nie jest możliwe oraz żadne hasło nie będzie pasowało, gdyż zostało ono zablokowane.
3. Pole trzecie zawiera listę nazw użytkowników systemu oddzielonych przecinkami, którzy mają prawo do administrowania grupą. W tym momencie jasne staje się występowanie hasła grupowego. Otóż administrator systemu tworzy nową grupę w systemie. Aby ograniczyć swoje czynności administratorskie wyznacza administratora (jednego lub kilku) dla tej grupy. Mogą oni dopisywać lub usuwać z niej użytkowników. Wykonanie tych czynności wymaga jednak znajomości hasła.
4. Pole czwarte zawiera powtórzenie zawartości czwartego pola linii z pliku */etc/group*. Znajdują się w nim oddzielone przecinkami, nazwy użytkowników w systemie.

## 2.2 Zarządzanie użytkownikami w dystrybucji Fedora

Podstawowymi komendami, służącymi do zarządzania użytkownikami w dystrybucji Fedora są:

- *useradd* - dodawanie użytkownika do systemu.
- *passwd* - zarządzanie hasłami użytkowników.

- *usermod* - zmiana atrybutów użytkownika (przynależności do grup, UID, katalogu domowego).
- *chage* - ustalanie minimalnego i maksymalnego okresu ważności hasła oraz ustalanie dnia ostatniej zmiany hasła.
- *userdel* - usuwanie użytkownika z systemu.
- *chfn* - zmiana opisu użytkownika (zwartości piątej kolumny w pliku */etc/passwd*).
- *chsh* - zmiana podstawowego interpretera użytkownika.

### 2.2.1 Dodawanie użytkownika do systemu

Użytkownika do systemu dodaje jego administrator. W większości systemów jest to użytkownik *root*, choć w niektórych przypadkach w systemie jest utworzona grupa użytkowników, których zadaniem jest zarządzanie użytkownikami. W dystrybucji Fedora użytkownika do systemu dodajemy używając komendy *useradd*. Komenda uruchomiona bez opcji i argumentów wypisuje krótką pomoc. Użycie komendy wymaga znajomości nazwy użytkownika (*login name*). Nazwa użytkownika powinna być zbudowana z liter i cyfr, ewentualnie zawierać znak kropki lub podkreślnika. Nie stosujemy znaków specjalnych takich, jak np. *#*, *\** czy *:*, jak również nazw użytkowników zbudowanych z samych cyfr. Wartości pozostałych atrybutów użytkownika zostaną nadane jako domyślne, odczytane z plików konfiguracyjnych z których komenda korzysta (w ten sposób ustala się np. ścieżkę dostępu do katalogu domowego użytkownika, przypisuje mu interpreter logujący) lub wyznaczone na ich podstawie (np. numer identyfikacyjny użytkownika). Użycie komendy *useradd* w ten sposób przedstawiono poniżej.

```
1 [root@messy ~]# useradd antek
```

Jej wykonanie spowoduje, że do systemu zostanie dodany użytkownik o nazwie *antek*, a wartości wszystkich jego atrybutów będą domyślne. W podstawowych plikach konfiguracyjnych pojawią się następujące linie:

- plik */etc/passwd*

```
1 antek:x:502:502::/home/antek:/bin/bash
```

- plik */etc/shadow*

```
1 antek:!!:14274:0:99999:7:::
```

- plik */etc/group*

```
1 antek:x:502:
```

- plik */etc/gpasswd*

```
1 antek:!::
```

Porównując dodane linie z fragmentami plików, które przedstawiono opisując ich składnię możemy stwierdzić, że numery identyfikacyjne nadane zostały według zasady: poprzednio nadany powiększony o 1. Wszyscy dodawani do systemu użytkownicy otrzymują jako podstawowy interpreter poleceń `/bin/bash`, nazwa katalogu domowego jest tożsama z nazwą użytkownika i znajduje się on w katalogu `/home`. W systemie została automatycznie utworzona grupa o nazwie takiej, jak dodany użytkownik. Jest ona grupą podstawową dla tego użytkownika, a jej numer identyfikacyjny pokrywa się z numerem identyfikacyjnym użytkownika.

Wartości atrybutów użytkownika mogą zostać ustalone podczas dodawania go do systemu dzięki wykożystaniu opcji komendy `useradd`. Do podstawowych należą:

- **-c opis** – pozwala ustalić zawartość piątej kolumny pliku `/etc/passwd`, czyli opis użytkownika.
- **-d katalog\_domowy** – służy do zdefiniowania ścieżki dostępu do katalogu domowego użytkownika.
- **-e data\_wygaśnięcia\_konta** – umożliwia ustalenie daty wygaśnięcia ważności konta, która stanowi zawartość ósmego pola pliku `/etc/shadow`. Podaje się ją w formacie `rrrr-mm-dd`.
- **-f liczba\_dni\_do\_blokady\_konta** – nadaje wartość w siódmym polu pliku `/etc/shadow`, w którym przechowywana jest liczba dni liczona od momentu wygaśnięcia hasła, po upływie których konto jest blokowane, jeśli użytkownik w tym czasie nie podłączał się do systemu.
- **-g numer\_identyfikacyjny\_grupy** – pozwala określić numer grupy podstawowej nowo dodawanego użytkownika.
- **-G numer\_identyfikacyjny\_grupy[, [numer\_identyfikacyjny\_grupy], ...]** – definiuje przynależność do grup dodatkowych.
- **-M** – blokuje tworzenie katalogu domowego dla dodawanego użytkownika, jeśli ustawienia domniemane komendy `useradd` je wymusza.
- **-m** – wymusza tworzenie katalogu domowego dla dodawanego użytkownika, jeśli ustawienia domniemane komendy `useradd` je blokuje. Do katalogu domowego zostaną skopiowane pliki konfiguracyjne, których wzorce znajdują się w katalogu `/etc/skel`.
- **-N** – użycie tej opcji powoduje, iż w trakcie dodawania użytkownika do systemu nie zostanie utworzona nowa grupa użytkowników o nazwie identycznej z nazwą użytkownika. W celu określenia grupy podstawowej należy użyć opcji `-g`. W przeciwnym wypadku grupą podstawową zostanie grupa określona w pliku `/etc/default/useradd`.
- **-o** – pozwala na dodanie użytkownika do systemu z numerem identyfikacyjnym, który posiada inny użytkownik (powtarzającym się).
- **-p zaszyfrowana\_postać\_hasła** – umożliwia ustawienie hasła użytkownika. Hasło musi zostać podane w postaci zaszyfrowanej w formacie `$identyfikator$sól$hasło`.
- **-s interpreter\_poleceń** – pozwala zdefiniować ścieżkę dostępu do podstawowego interpretera poleceń użytkownika.
- **-u numer\_identyfikacyjny\_użytkownika** – umożliwia nadanie numeru identyfikacyjnego dodawanemu użytkownikowi.

W opisie komendy *useradd* pojawiły się pliki i katalogi z zawartości których korzysta ona podczas dodawania użytkownika do systemu. Są to pliki */etc/default/useradd*, */etc/login.defs* oraz katalog */etc/skel*.

Zawartość pliku */etc/default/useradd* przedstawiono poniżej.

```

1 # useradd defaults file
2 GROUP=100
3 HOME=/home
4 INACTIVE=-1
5 EXPIRE=
6 SHELL=/bin/bash
7 SKEL=/etc/skel
8 CREATE_MAIL_SPOOL=yes

```

Znajdują się w nim wartości podstawowych atrybutów użytkownika istotnych bardziej dla użytkownika niż sposobu działania czy bezpieczeństwa systemu. Przykładowo katalog, w którym będą zakładane katalogi domowe użytkowników (*HOME*, ścieżka dostępu do podstawowego interpretera poleceń (*SHELL*, czy ścieżka dostępu do podstawowych plików definiujących środowisko pracy użytkownika (*SKEL*).

Zawartość pliku */etc/login.defs* to wartości zmiennych, które mają wpływ na sposób pracy samej komendy *useradd*, jak również wartości atrybutów użytkownika, które mają wpływ na bezpieczeństwo systemu. Przykład pliku przedstawiono poniżej.

```

1 # Directory where mailboxes reside, _or_ name of file, relative to the
2 # home directory. If you _do_ define both, MAIL_DIR takes precedence.
3 # QMAIL_DIR is for Qmail
4 #
5 #QMAIL_DIR      Maildir
6 MAIL_DIR        /var/spool/mail
7 #MAIL_FILE      .mail
8
9 # Password aging controls:
10 #
11 #      PASS_MAX_DAYS      Maximum number of days a password may be used.
12 #      PASS_MIN_DAYS      Minimum number of days allowed between password changes.
13 #      PASS_MIN_LEN        Minimum acceptable password length.
14 #      PASS_WARN_AGE      Number of days warning given before a password expires.
15 #
16 PASS_MAX_DAYS    99999
17 PASS_MIN_DAYS    0
18 PASS_MIN_LEN     5
19 PASS_WARN_AGE    7
20
21 #
22 # Min/max values for automatic uid selection in useradd
23 #
24 UID_MIN          500
25 UID_MAX          60000
26

```



```
27 #
28 #
29 # Min/max values for automatic gid selection in groupadd
30 #
31 GID_MIN          500
32 GID_MAX          60000
33
34 #
35 # If defined, this command is run when removing a user.
36 # It should remove any at/cron/print jobs etc. owned by
37 # the user to be removed (passed as the first argument).
38 #
39 #USERDEL_CMD      /usr/sbin/userdel_local
40
41 #
42 # If useradd should create home directories for users by default
43 # On RH systems, we do. This option is overridden with the -m flag on
44 # useradd command line.
45 #
46 CREATE_HOME      yes
47
48 # The permission mask is initialized to this value. If not specified,
49 # the permission mask will be initialized to 022.
50 UMASK            077
51
52 # This enables userdel to remove user groups if no members exist.
53 #
54 USERGROUPS_ENAB yes
55
56 # Use MD5 or DES to encrypt password? Red Hat use MD5 by default.
57 MD5_CRYPT_ENAB no
58
59 ENCRYPT_METHOD SHA512
```

Jak widać, w pliku tym ustalamy wartości atrybutów decydujących o częstotliwości zmiany hasła (*PASS\_\**, linie 16–19), zakres wartości numeru identyfikacyjnego użytkownika (*UID\_MIN*, *UID\_MAX*, linie 24, 25), zakres wartości numeru identyfikacyjnego grupy (*GID\_MIN*, *GID\_MAX*, linie 30, 31) czy ścieżka dostępu do katalogu przechowującego pliki z pocztą użytkowników (*MAIL\_DIR*, linia 6). W pliku tym możemy również zdefiniować ścieżkę dostępu do pliku zawierającego program wykorzystywany podczas usuwania użytkownika z systemu (*USERDEL\_CMD* linia 38). Istotnymi z punktu widzenia bezpieczeństwa systemu są możliwości wyboru funkcji szyfrującej hasła. Rodzina systemów *Red Hat*, z której wywodzi się Fedora, do wersji 8 jako domniemanej używała funkcji MD5. Obecnie stosuje SHA512. W podanym przykładzie została ona zablokowana przez nadanie wartości *no* zmiennej *MD5\_CRYPT\_ENAB* (linia 56). Wyboru funkcji szyfrującej dokonano nadając wartość *SHA512* zmiennej *ENCRYPT\_METHOD* (linia 58). Ma on charakter globalny, to znaczy funkcją tą będą szyfrowane zarówno hasła użytkowników, jak i grupowe. Zśród praktycznych definicji należy wspomnieć o możliwości zdefiniowania grupy, która będzie podstawową dla każdego dodawanego użytkownika. Przykładowy wpis:

```
1  GROUP          staff
```

spowoduje, że unikniemy tworzenia nowej grupy przy okazji dodawania użytkownika do systemu, a grupą podstawową dla każdego będzie grupa *staff* chyba, że podczas uruchamiania komendy *useradd* użyto opcji *-g*.

Katalog */etc/skel* zawiera podstawowe pliki wykorzystywane przez interpretery poleceń do konfigurowania środowiska pracy użytkownika oraz pliki konfigurujące interfejs graficzny (KDE lub Gnome). Są one kopiowane przez komendę *useradd* do katalogu domowego użytkownika. Budowę oraz sposób wykorzystania plików przez interpretery omówiono szczegółowo w Rozdziale 5.3. Będąc administratorem możemy modyfikować istniejące pliki, jak również umieszczać w katalogu własne, np. wymagane przez zainstalowane w systemie aplikacje.

## 2.2.2 Zmiana konfiguracji użytkownika

### Aspekty związane z hasłem

Zwróćmy uwagę, że dodany do systemu komendą *useradd* użytkownik nie może się jeszcze do niego podłączyć, gdyż w drugim polu opisującej go w pliku */etc/shadow* linii znajdują się *!!!*. Oznacza to, że hasło jest zablokowane (pojawienie się na początku drugiego pola pliku */etc/shadow* oznacza zablokowanie hasła). Stąd kolejną czynnością po dodaniu jest ustalenie hasła. Czynność tą wykonuje administrator systemu, wykorzystując komendę *passwd* z argumentem będącym nazwą użytkownika. Komenda *passwd* użyta bez argumentu zmienia hasło użytkownikowi, który ją uruchomił. Proces nadania hasła zdefiniowanemu użytkownikowi *antek* przedstawiono poniżej.

```
1  [root@messy ~]# passwd antek
2  Changing password for user antek.
3  New UNIX password:
4  BAD PASSWORD: it is based on a dictionary word
5  Retype new UNIX password:
6  passwd: all authentication tokens updated successfully.
7  [root@messy ~]#
```

Zwróćmy uwagę na komunikat mówiący o błędzie wynikającym ze zbyt prostej postaci hasła. Komenda *passwd* dopuszcza jednak do stosowania tego hasła, gdyż ustawia je administrator. Objawia się to prośbą o podanie hasła ponownie w celu jego weryfikacji. Takie samo hasło zaproponowane przez zwykłego użytkownika zostałoby odrzucone. Postać zaszyfrowana została zapisana w odpowiednim miejscu, co przedstawia poniższa linia zaczerpnięta z pliku */etc/shadow* i od tej pory użytkownik może do systemu się podłączać.

```
1  antek:$6$gtFZzak8$0YsTdqcTKkRIf3IHQ1G3wTIIXQFMGQYHPKVSFZUTR
2  y9Q0Azh16xiUtojcRMrEGCJIezXrPco8.PJ7FPePrZbZ0:14278:0:99999:7:::
```

Do podstawowych opcji komendy *passwd* należą:

- **-l** – pozwala na blokowanie konta użytkownika, którego nazwa została użyta jako argument. Jest to realizowane przez wstawienie znaków *!!* przed zaszyfrowaną postacią hasła, w drugim polu linii definiującej użytkownika w pliku */etc/shadow*. Wstawiana jest para znaków *!!*,

stąd wielokrotne użycie komendy z tą opcją w odniesieniu do tego samego użytkownika nie przynosi żadnych zmian. Opcję może wykorzystywać jedynie użytkownik *root*. Dla pozostałych jest niedostępna.

- **-u** – działanie odwrotne do opcji **-l**. Pozwala odblokować konto użytkownika. Fizycznie sprowadza się to do usunięcia znaków !! sprzed postaci zaszyfrowanej hasła, zapisanego w drugim polu linii opisującej użytkownika w pliku */etc/shadow*. Opcja jest dostępna jedynie dla użytkownika *root*.
- **-d** – użycie tej opcji z argumentem będącym nazwą użytkownika powoduje, że użytkownik ten przy podłączaniu się do systemu nie będzie musiał wykorzystywać hasła. W praktyce będzie miał ustawione hasło puste, co odpowiada pustemu drugiemu polu w linii opisującej użytkownika w pliku */etc/shadow*. Opcję tę może wykorzystywać jedynie użytkownik *root*.
- **-n *liczba\_dni*** – opcja pozwala ustawić minimalny czas ważności hasła licząc w dniach od momentu ostatniej zmiany hasła. Zabezpiecza więc przed zbyt częstymi zmianami hasła. Odpowiednia wartość pojawia się w czwartym polu linii opisującej użytkownika w pliku */etc/shadow*. Opcja dostępna tylko dla użytkownika *root*.
- **-x *liczba\_dni*** – wykorzystując tę opcję ustalamy maksymalny wiek ważności hasła licząc w dniach od momentu ostatniej zmiany hasła. W ten sposób wymuszamy odpowiednią częstotliwość zmiany hasła. Informacja zostanie zapisana w piątym polu linii opisującej użytkownika w pliku */etc/shadow*. Opcję może wykorzystywać użytkownik *root*.
- **-w *liczba\_dni*** – liczba dni, które dzielą użytkownika od utraty ważności hasła, przez które będzie on ostrzegany o zbliżającej się konieczności jego zmiany. Wartość ta jest przechowywana w szóstym polu linii opisującej użytkownika w pliku */etc/shadow*. Opcja dostępna tylko dla użytkownika *root*.

Ostatnią czynnością wykonywaną podczas dodawania użytkownika do systemu będzie zatem nałożenie ograniczeń czasowych na hasło w postaci minimalnego i maksymalnego okresu ważności hasła. Dodatkowo możemy zdefiniować liczbę dni przez które będzie on ostrzegany o zbliżającym się terminie zmiany hasła. Wartości są zależne od prowadzonej polityki bezpieczeństwa. Maksymalny okres ważności hasła jest wybierany z zakresu od 28 do 360 dni. Wartości mniejsze dotyczą systemów bardziej restrykcyjnych. Minimalny okres ważności hasła należy ustalić w oparciu o wartość maksymalnego okresu ważności oraz długość listy haseł historycznych. Lista ta działa na zasadzie kolejki FIFO i zawiera hasła ostatnio używane, które nie mogą być aktualnie ustawione. Jeśli przykładowo długość listy ustalimy na 3, a minimalny okres ważności hasła na 7 dni, to wówczas to samo hasło może pojawić się dopiero po 21 dniach. Dla ustawień bardziej restrykcyjnych wydłuża się listę historii haseł do 5, a zmniejsza minimalny okres ważności hasła do 3 dni. W systemach, w których stosuje się ustawienia mniej restrykcyjne skraca się listę, a wydłuża minimalny okres ważności hasła lub wręcz rezygnuje z tego mechanizmu.

Ograniczenia wprowadzamy następującą komendą:

```
1 [root@messy ~]# passwd -n 5 -x 28 -w 3 antek
2 Adjusting aging data for user antek.
3 passwd: Success
4 root@messy ~]#
```

Modyfikuje ona linię opisującą użytkownika w pliku */etc/shadow* do postaci:

```

1 antek:$6$gtFZzak8$0YsTdqcTKkRIf3IHq1G3wTIIXQFMGQYHPKVSFZUTR
2 y9Q0Azh16xiUtojcRMrEGCJIezXrPco8.PJ7FPePrZbZ0:14278:5:28:3:::

```

Efektywne działanie ograniczania minimalnego okresu ważności hasła wymaga jeszcze zdefiniowania długości listy ostatnio wykorzystywanych haseł. Realizuje się to wykorzystując mechanizmy PAM, które zostały omówione w podrozdziale 2.9, a przykład omówiony w punkcie 2.9.3 dotyczy tego właśnie zagadnienia.

Dobłą praktyką jest potraktowanie ustawionego przez administratora hasła jako tymczasowego i wymuszenie na użytkowniku jego zmiany podczas pierwszego podłączenia się do systemu. Jak wspomniano, wystarczy, aby w trzeciej kolumnie linii opisującej użytkownika w pliku */etc/shadow* pojawiła się wartość 0, a proces logujący wymusi na użytkowniku zmianę hasła przy pierwszym podłączeniu do systemu. Wymaga to użycia komendy *chage* w następującej postaci:

```

1 [root@messy etc]# chage -d 0 antek

```

Komendę *chage* omówiono w następnym punkcie.

Po przeczytaniu opisu opcji komendy *passwd* nasuwa się proste pytanie: czy posiada ona opcje dostępne dla zwykłego użytkownika? Nie. Zwykły użytkownik przy pomocy tej komendy może zmienić sobie hasło i to pod warunkiem, że nie zapomniał obowiązującego. Jeśli zapomniał hasła, to nowe może ustawić użytkownik *root*, którego komenda *passwd* nie zapyta o aktualne. Jak wspomniano, hasła w systemie nie można podejrzeć, gdyż przechowywane jest jedynie w postaci zaszyfrowanej. Co jeśli hasła zapomniał użytkownik *root*? Procedur postępowania jest wiele i zależą one ściśle od dystrybucji systemu.

### Zmiana pozostałych atrybutów użytkownika

Przyjmując za kryterium podziału częstość zmian atrybutów możemy je podzielić na te, które zmieniamy często, zmieniane sporadycznie oraz takie, które przez cały czas w którym użytkownik jest zdefiniowany w systemie nie zmieniają swojej wartości. Pierwsza grupa obejmuje atrybuty mające przede wszystkim wpływ na bezpieczeństwo systemu. Należy do nich hasło użytkownika. Podstawy administrowania nim już poznaliśmy. Stosunkowo rzadko modyfikacji podlega przynależność użytkowników do grup. Te zmiany najlepiej zrealizować komendą *usermod*. Uprawnienia do zmian posiada w tym wypadku administrator systemu. Praktyka pokazuje, że jeszcze rzadziej zmieniany jest opis użytkownika zawarty w piątym polu opisującej go linii w pliku */etc/passwd*. Sporadycznie zmianie podlega ścieżka dostępu do podstawowego interpretera poleceń. Wartości dwóch ostatnich atrybutów mogą zostać zmienione przez administratora systemu odpowiednio komendami *usermod* lub *chfn* i *chsh*. Dwie ostatnie komendy może również wykonać zwykły użytkownik systemu. Ich dostępność w systemie nie jest pożądana z uwagi na specyficzne prawa dostępu, co zostało dokładnie omówione w Rozdziale 3. Spośród już raczej teoretycznych możliwości zmian odnotować należy zmianę numeru identyfikacyjnego użytkownika oraz nazwy katalogu domowego. Ścieżka dostępu do katalogu domowego może ulec zmianie jeśli katalog, w którym przechowywane są katalogi domowe użytkowników przenosimy w inne miejsce drzewa katalogów. Zmianę wartości tych atrybutów użytkownika najprościej zrealizować komendą *usermod*. Również tą komendą załatwimy zmianę nazwy użytkownika w systemie. Ma ona miejsce najczęściej wówczas, gdy nazwa użytkownika w systemie po prostu nie podoba się jej właścicielowi. Jednak działania te mają charakter jednokrotny w stosunku do danego użytkownika i występują zazwyczaj przed rozpoczęciem jego aktywności w systemie. Stąd należy również rozważyć, czy nie prościej będzie danego użytkownika z systemu usunąć i utworzyć nowego o właściwej nazwie.

Zobaczmy jakie opcje posiadają rozważane komendy. Zaczniemy od komendy *usermod*. Uru-  
chamiając ją musimy pamiętać o podaniu jako argumentu nazwy użytkownika w systemie,  
którego wartości atrybutów będziemy modyfikować. Możliwości komendy określają następujące  
opcje:

- **-c opis** – pozwala zmienić opis użytkownika w systemie przechowywany w piątym polu linii  
opisującej użytkownika w pliku */etc/passwd*. Jeśli opis składa się z kilku ciągu znaków (np.  
nazwisko i imię) oddzielonych znakiem białym (spacja lub tabulacja), to należy umieścić  
go między znakami cudzysłowia lub apostrofu (np. *chmod -c "KowalskiJan" jan*).
- **-d katalog\_domowy** – zmiana ścieżki dostępu do katalogu domowego. Jeśli użyto dodatkowo  
opcji **-m**, to przenoszona jest zawartość dotychczasowego katalogu domowego.
- **-e data\_ważności\_konta** – dla kont tymczasowych, umożliwia określenie okresu ważności  
konta. Informacja ta jest przechowywana w ósmym polu linii opisującej użytkownika w  
pliku */etc/shadow*. Datę podaje się w formacie rrrr-mm-dd. Wprowadzenie pustego napisu  
oznacza, że nie zostanie nigdy zablokowane.
- **-f liczba\_dni** – użycie komendy *usermod* z tą opcją pozwala na określenie liczby dni, po  
upływie których licząc od momentu wygaśnięcia ważności hasła konto użytkownika zostanie  
automatycznie zablokowane. Wartość 0 oznacza, że nastąpi to z chwilą utraty ważności  
przez hasło, pusty napis, że nie zostało wprowadzone żadne ograniczenie.
- **-g grupa** – pozwala na zmianę grupy podstawowej użytkownika na podaną jako wartość  
opcji. Grupa może zostać określona numerem identyfikacyjnym grupy w systemie lub jej  
nazwą.
- **-G grupa1[, grupa2], ...** – umożliwia zdefiniowanie listy grup, które są dodatkowe dla użyt-  
kownika. Opcja działa w trybie „nadpisywania”, tzn. usuwa poprzednią definicję przyna-  
leżności do grup i ustawia podaną w wartości opcji. Jeśli chcemy zachować dotychczasową  
informację o przynależności do grup dodatkowych i dopisać nową grupę, to wówczas musi  
użyć dodatkowo opcji **-a**.
- **-l nazwa\_użytkownika** – opcja ta pozwala na zmianę nazwy użytkownika w systemie. Nie  
ulega zmianie nazwa katalogu domowego.
- **-L** – blokuje hasło użytkownika. Operacja polega na wstawieniu przed postacią zaszyfro-  
waną hasła, w drugim polu linii opisującej użytkownika w pliku */etc/shadow* znaku **!**.  
Wielokrotne wykonanie komendy nie dopisuje kolejnych znaków, stąd odblokowanie wyko-  
nujemy jednokrotnie.
- **-p hasło** – umożliwia ustawienie hasła użytkownika. Wymagane jest podanie postaci za-  
szyfrowanej hasła w formacie *\$identyfikator\$sól\$hasło*.
- **-u numer\_identyfikacyjny** – zmienia numer identyfikacyjny użytkownika na podany jako  
wartość opcji **-p**. Numer powinien być niepowtarzalny. Należy zaznaczyć, iż komenda zmie-  
nia także numer identyfikacyjny właściciela indywidualnego każdego pliku w katalogu oso-  
bistym użytkownika, którego nazwa została podana jako argument wywołania komendy.
- **-U** – pozwala na odblokowanie hasła. W praktyce polega to na usunięciu znaku **!**, który  
znajduje się przed postacią zaszyfrowaną hasła w drugim polu linii opisującej użytkownika  
w pliku */etc/shadow*.

Przeglądając opcje komendy *usermod* nasuwa się wniosek, iż z jej pomocą możemy zmienić wartość każdego atrybutu opisującego użytkownika w systemie, która została ustalona podczas dodawania użytkownika do systemu. Omówmy jeszcze dwie komendy, które służą do modyfikowania informacji o użytkowniku oraz ścieżki dostępu do podstawowego interpretera poleceń. Są to odpowiednio *chfn* oraz *chsh*. Obie mogą zostać uruchomione na dwa sposoby. Pierwszy, z wykorzystaniem opcji, wymaga podania w linii komend wartości, które mają zostać nadane odpowiednim atrybutom. Jeśli natomiast komenda zostanie uruchomiona bez opcji, to przechodzi do trybu interaktywnego, w którym „odpytuje” o wartości kolejnych atrybutów. Dotychczasowe wartości są wykorzystywane jako domyślne i akceptowane klawiszem *Enter*. Argumentem dla obu komend jest nazwa użytkownika w systemie, którego zmiany dotyczą. Jeśli komendę uruchamia zwykły użytkownik, to nie podaje on argumentu, a zmiany dotyczą wówczas bieżącego użytkownika. Wynika to z faktu, że nie ma on uprawnień do modyfikowania wartości atrybutów innego użytkownika. Użytkownik *root* dokonuje modyfikacji wartości atrybutów innych użytkowników, podając ich nazwy jako argumenty. Ze względów bezpieczeństwa, komenda uruchomiona przez zwykłego użytkownika, przed rozpoczęciem działania prosi o podanie jego hasła.

Podstawowe opcje komendy *chfn* to:

- **-f** *opis\_użytkownika* – pozwala na wprowadzenie opisu użytkownika. Najczęściej są to imię i nazwisko.
- **-o** *pokój\_służbowy* – umożliwia podanie miejsca pracy użytkownika. Może to być na przykład numer pokoju lub inna informacja pozwalająca zlokalizować użytkownika.
- **-p** *numer\_telefonu* – pozwala ustalić numer telefonu służbowego użytkownika.
- **-h** *numer\_telefonu* – pozwala ustalić numer telefonu domowego użytkownika.

Jeśli podczas uruchomienia komendy *chfn* nie wykorzystujemy opcji, to przebieg jej działania jest następujący:

```

1  [jan@messy ~]$ chfn
2  Changing finger information for jan.
3  Password:
4  Name []: Jan Kowalski
5  Office []: C2 pok.433
6  Office Phone []: 6179999
7  Home Phone []: +0122345678
8
9  Finger information changed.
10 [jan@messy ~]$
```

Komendę uruchomił zwykły użytkownik, stąd pytanie o hasło. Wartości domyślne są podawane w nawiasach kwadratowych. Ponieważ przed wykonaniem komendy zawartość piątego pola linii opisującej użytkownika *jan* w pliku */etc/passwd* była pusta, stąd brak odpowiedzi. Po jej wykonaniu linia opisująca użytkownika w pliku */etc/passwd* jest następująca:

```

1  jan:x:501:501:Jan Kowalski,C2 pok.433,6179999,+0122345678:/home/jan:/bin/bash
```

Jak widać piąte pole zostało podzielone na 4 fragmenty zawierające wprowadzoną informację. Komenda *finger* zwraca informacje w następujący sposób:

```

1 [root@messy ~]# finger jan
2 Login: jan                               Name: Jan Kowalski
3 Directory: /home/jan                     Shell: /bin/bash
4 Office: C2 pok.433, 617-9999             Home Phone: +0122345678
5 On since Thu Feb  5 18:13 (CET) on pts/1 from thorin.icsr.agh.edu.pl
6      4 minutes 7 seconds idle
7 No mail.
8 No Plan.
```

Na zakończenie komenda *chsh* pozwalająca zmienić podstawowy interpreter poleceń. Posiada ona dwie użyteczne opcje. Są to:

- **-l** – dostarcza listę dostępnych w systemie interpreterów poleceń. Jest ona zapisana w pliku */etc/shells*.
- **-s** *ścieżka\_dostępu\_do\_interpretera* – umożliwia zmianę podstawowego interpretera poleceń na ten, do którego ścieżka dostępu jest wartością opcji.

Występowanie w systemie interpretera poleceń, na który pokazuje ścieżka dostępu zapisana w ostatnim polu linii w pliku */etc/passwd* ma wpływ na możliwość podłączania się do systemu oraz na bezpieczeństwo systemu. Istnieją metody włamywania się do systemu polegające na wykorzystaniu zamiast interpretera poleceń specjalnego programu. Stąd komenda *chsh* nie zmieni interpretera poleceń na taki, który nie występuje w pliku */etc/shells*. Prawo modyfikacji tego pliku ma użytkownik *root*. Stąd dla podniesienia poziomu bezpieczeństwa systemu uniemożliwia się uruchomienie komendy *chsh* zwykłym użytkownikom lub wręcz usuwa ją z systemu. Zmianę interpretera poleceń dokonuje wówczas administrator systemu zmieniając edytorem zawartość pliku */etc/passwd*.

Przed przystąpieniem do zmiany interpretera poleceń sprawdźmy, jakie interpretery są dostępne w naszym systemie. Warto w tym miejscu nadmienić, że interpreter poleceń jest niezależny od dystrybucji systemu. Dostępnych jest wiele interpreterów różniących się chociażby sposobem interpretacji znaków specjalnych, listą poleceń wbudowanych czy składnią dostępnego w nich języka programowania. Każdy z nich ma swoich zagorzałych zwolenników i przeciwników. Stąd warto mieć w systemie dostępnych kilka, a jeśli okaże się, że jakiegoś brakuje, to można go w każdej chwili doinstalować. Zmianę interpretera poleceń rozpoczynamy od ustalenia możliwości:

```

1 [jan@messy ~]$ chsh -l
2 /bin/sh
3 /bin/bash
4 /sbin/nologin
5 /bin/zsh
6 [jan@messy ~]$
```

Proces zmiany wygląda następująco:

```

1 [jan@messy ~]$ chsh
2 Changing shell for jan.
3 Password:
4 New shell [/bin/bash]: /bin/sh
```

```

5 Shell changed.
6 [jan@messy ~]$

```

W tym przypadku, w nawiasach kwadratowych pojawiła się podpowiedź, która jest ścieżką dostępu do aktualnego interpretera poleceń. Jeśli podana zostanie ścieżka dostępu do interpretera poleceń, która nie jest wyszczególniona w pliku `/etc/shells`, to nie zostanie ona ustawiona:

```

1 [jan@messy ~]$ chsh
2 Changing shell for jan.
3 Password:
4 New shell [/bin/sh]: /bin/tcsh
5 chsh: "/bin/tcsh" does not exist.
6 [jan@messy ~]$

```

### 2.2.3 Monitorowanie podłączania się użytkowników do systemu

Zagadnienie to można rozpatrywać w odniesieniu do użytkowników, którzy aktualnie są do systemu podłączeni oraz historii ich podłączania się do systemu. W pierwszym przypadku z pomocą przychodzą nam dwie komendy, a mianowicie *w* oraz *who*. W drugim przypadku mamy do dyspozycji komendę *last*. Mogą one być uruchamiane przez każdego użytkownika systemu. Administrator systemu może również wykorzystywać informacje, która jest zapisywana w dziennikach systemu. Są to pliki przechowywane w katalogu `/var/log`.

#### Komenda *who*

Komenda *who* uruchomiona bez argumentu dostarcza informacji, które czerpie z pliku `/var/run/utmp`. Użycie argumentu pozwala wskazać ewentualnie na inny plik. Komenda użyta bez opcji wypisuje listę użytkowników aktualnie podłączonych do systemu, przy czym każda sesja opisana jest pojedynczą linią. Przykład zamieszczono poniżej:

```

1 [root@messy ~]# who
2 jan      tty1      2009-02-05 19:54
3 root     tty2      2009-02-05 17:04
4 jan      pts/1      2009-02-05 18:13 (ps3.icsx.agh.edu.pl)
5 antek    pts/2      2009-02-05 19:13 (141.152.99.13)
6 [root@messy ~]#

```

Każda linia jest podzielona na 4 kolumny:

1. W pierwszej kolumnie znajduje się nazwa użytkownika w systemie.
2. Druga linia zawiera opis terminala. Jeśli użytkownik jest podłączony lokalnie z konsoli, to w kolumnie tej znajduje się opis w postaci **tty***x*, gdzie *x* oznacza numer konsoli. Jeśli natomiast użytkownik jest podłączony zdalnie, to wówczas sesja wykorzystuje tzw. pseudoterminal. Stąd opis w postaci **pts**/*x*, gdzie *x* oznacza w tym przypadku numer pseudoterminala.
3. W trzeciej kolumnie znajduje się data podłączenia się do systemu w formacie *rok-miesiąc-dzień godzina:minuta*.



4. Zawartość czwartej kolumny zależy od sposobu podłączenia się użytkownika do systemu. Jeśli jest on podłączony lokalnie, to kolumna ta jest pusta. W przypadku podłączenia zdalnego, w kolumnie tej pojawia się adres symboliczny lub adres IP hosta, z którego nawiązana jest sesja.

Stosując opcje, dzięki komendzie *w* możemy uzyskać także inne informacje. Do podstawowych opcji należą:

- **-H** – w pierwszej linii wypisywany jest nagłówek opisujący zawartość kolumn.
- **-T** – wypisuje dodatkową informację o stanie terminala sesji, a konkretnie czy użytkownik dopuszcza, czy też nie wypisywanie komunikatów wysyłanych przez innych użytkowników (np. przy pomocy komendy *write*). Oznaczenia stanu mogą być następujące: + - dopuszcza się wypisywanie, - wypisywanie nie jest dopuszczone, ? określenie stanu nie jest możliwe (np. uruchomiony został interfejs graficzny). Dopuszczanie lub nie pojawiania się komunikatów ustala się przy pomocy komendy *mesg*, użytej ze słowem kluczowym odpowiednio *yes* lub *no*. Wynik działania opcji **-T** oraz **-H** jest następujący:

```

1 [root@messy ~]# who -TH
2 NAME      LINE      TIME      COMMENT
3 root      + tty1      2013-08-12 15:00
4 wacek     ? tty3      2013-08-12 15:04
5 jan       - tty2      2013-08-12 15:01

```

- **-b** – wypisuje czas ostatniej inicjalizacji systemu.
- **-q** – wypisuje nazwy użytkowników podłączonych do systemu oraz ich liczbę. Informacja ma postać:

```

1 [jan@messy ~]$ who -q
2 jan root root antek
3 # users=4

```

- **-r** – pozwala uzyskać informację o aktualnym poziomie pracy systemu oraz czasie jego inicjalizacji, w formacie jak poniżej:

```

1 [root@messy ~]# who -r
2          run-level 3  2013-08-12 14:46          last=

```

- **-s** – jest opcją domniemaną. Komenda dostarcza informacje w formacie takim, jak w przypadku użycia jej bez opcji.
- **-u** – opcja ta dostarcza informację w formacie takim jak w przypadku użycia komendy bez opcji, ale poszerzoną o czas nieaktywności użytkownika (rozumianej jako przedział czasu, który minął od momentu wydania ostatniej komendy) oraz numeru identyfikacyjnego procesu dzięki któremu użytkownik jest podłączony do systemu. Stanowią one czwartą oraz piątą kolumnę w przedstawionym poniżej przykładzie:

```

1 [jan@messy ~]$ who -u
2 jan      tty2      2009-02-06 13:37 00:04      2032

```

```

3 root    tty1      2009-02-06 13:37 00:04      2034
4 root    pts/0      2009-02-06 13:00 00:10      2080 (ps3.icsx.agh.edu.pl)
5 jan     pts/1      2009-02-06 13:00 .          2158 (ps3.icsx.agh.edu.pl)
6 [jan@messy ~]$

```

Domyślnym argumentem komendy *who* jest plik *wtmp* znajdujący się w katalogu */var/log*. Rekordy zapisane w tym pliku przechowują informacje o podłączeniach i zamknięciach sesji użytkowników, inicjalizacjach i restartach systemu. Ze względu na szybko rosnący rozmiar pliku, system dokonuje cyklicznych kopiowań jego zawartości do plików archiwalnych, znajdujących się w katalogu */var/log* o nazwach *wtmp.* z dodaną cyfrą od 1 do 6. Plik */var/log/wtmp.6* zawiera zdarzenia najstarsze, zaś */var/log/wtmp.1* zdarzenia najmłodsze. Fragment historii podłączeń przedstawiono poniżej.

### Komenda *w*

Komenda *w* dostarcza informacji nieco bardziej szczegółowych. Pozwala dowiedzieć się, kto jest do systemu podłączony, jak mocno wykorzystuje podstawowy zasób bierny, jakim jest procesor oraz jakie jest ogólne obciążenie systemu. Użycie bez argumentów dostarcza informacji w postaci:

```

1 [jan@messy ~]$ w
2 14:18:29 up 1:45, 4 users, load average: 0.00, 0.00, 0.00
3 USER      TTY      FROM          LOGIN@  IDLE   JCPU   PCPU   WHAT
4 jan       tty2      -              13:37   40:57   0.07s   0.07s  -sh
5 root      tty1      -              13:37   41:02   0.08s   0.08s  -bash
6 root      pts/0     thorin.icsr.agh. 13:00   47:03   0.13s   0.01s  man who
7 jan       pts/1     thorin.icsr.agh. 13:00   0.00s   0.24s   0.01s  w
8 [jan@messy ~]$

```

Pierwsza linia zawiera informacje o ogólnym stanie systemu. Składa się z trzech pól oddzielonych przecinkiem. Pola oznaczają:

1. Bieżący czas systemowy. Po słowie *up* pojawia się informacja jak długo system jest aktywny, a mówiąc prościej ile czasu minęło od jego inicjalizacji. Format, to *godziny:minuty*, lub *dni:godziny*, zależnie od czasu aktywności systemu. Jeśli miał on miejsce wcześniej niż 24 godziny temu, to jest on podawany w dniach i godzinach.
2. W drugiej kolumnie znajduje się informacja o liczbie użytkowników aktualnie podłączonych do systemu.
3. Trzecia kolumna, to średnie obciążenie systemu kolejno od lewej, z ostatniej minuty, 5-ciu i 15-tu minut. Jest to średnia długość kolejki procesów gotowych do wykonania (oczekujących na procesor). We współczesnych systemach komputerowych, zauważalny spadek wydajności można zaobserwować, jeśli wartości te są większe od 5. Informacja ta pochodzi z pliku tekstowego */proc/loadavg*.

Druga linia to nagłówek tabeli będący opisem zawartości kolumn. W kolumnach znajduje się:

- *USER* – nazwa użytkownika w systemie.
- *TTY* – opis terminala przy pomocy którego użytkownik jest podłączony w systemie.

- *FROM* – jeśli użytkownik jest podłączony zdalnie, to w tej kolumnie pojawia się adres IP lub symboliczny hosta, z którego przebiega sesja. Dla połączeń lokalnych kolumna nie zawiera żadnych informacji.
- *LOGIN@* – czas podłączenia się do systemu.
- *IDLE* – okres czasu, który minął od wydania przez użytkownika ostatniej komendy, określany czasem bezczynności użytkownika w systemie.
- *JCPU* – to czas, w którym procesor wykonywał instrukcje wszystkich procesów skojarzonych z terminalem opisanym w kolumnie *TTY*.
- *PCPU* – to czas poświęcony przez procesor na wykonywanie instrukcji procesu opisanego w kolumnie *WHAT*.
- *WHAT* – nazwa aktualnie wykonywanego procesu.

Działanie opcji komendy *w* ogranicza się jedynie do prostego formatowania dostarczanej informacji. Użycie opcji **-h** powoduje, że linia nagłówka tabeli nie zostanie wypisana. Opcja **-s** sprawia, że nie zostaną wypisane kolumny zawierające czas podłączenia się do systemu (*LOGIN@*) oraz kolumny *JCPU* i *PCPU*.

### Komenda *last*

Komendy *who* oraz *w* dostarczają informacji o stanie bieżącym systemu. Historię podłączanie się użytkowników do systemu dostarcza komenda *last*. Informacje czerpie z binarnego pliku */var/log/wtmp*. Plik przeglądany jest od końca, więc listę otwiera informację o zdarzeniach najbliższych nam czasowo. Komenda użyta bez opcji wypisuje całą zawartość pliku. Możemy ją ograniczyć do ściśle określonej liczby linii podając tę wartość po opcji **-n**. Przykładowo:

```

1 [jan@messy ~]$ last -n 10
2 jan      tty2                      Fri Feb  6 13:37    still logged in
3 root     tty1                      Fri Feb  6 13:37    still logged in
4 jan      pts/1                    sp3.icsx.agh. Fri Feb  6 13:00    still logged in
5 root     pts/0                    sp3.icsx.agh. Fri Feb  6 13:00    still logged in
6 reboot   system boot  2.6.25-14.fc9.i6 Fri Feb  6 12:33      (03:43)
7 jan      tty1                      Thu Feb  5 19:54 - down  (00:03)
8 jan      pts/1                    sp3.icsx.agh. Thu Feb  5 18:13 - 19:57 (01:44)
9 root     pts/0                    sp3.icsx.agh. Thu Feb  5 17:04 - 19:57 (02:53)
10 reboot   system boot  2.6.25-14.fc9.i6 Thu Feb  5 17:01      (02:56)
11 jan      pts/0                    sp3.icsx.agh. Wed Feb  4 14:52 - down  (03:37)
12
13 wtmp begins Tue Feb  3 16:36:40 2009
14 [jan@messy ~]$

```

Format dostarczanej informacji jest bardzo podobny do występującego w komendzie *w*. Jedyna różnica, to dodatkowa informacja o czasie trwania sesji. Mamy więc sesje trwające od konkretnej chwili czasowej, do konkretnej chwili czasowej z podaniem długości trwania w formacie *minuty:sekundy*. Są sesje zakończone przez zamknięcie systemu (*- down*). I w końcu sesje, które aktualnie trwają, opisane jako *still logged in*. Komenda *last* dostarcza również informacji o

inicjalizacji systemu. Zdarzenie te opisują linie, które w kolumnie zawierającej nazwę użytkownika mają słowo *reboot*. Opis terminala zawiera informacje *system boot*, a w kolumnie opisującej hosta podana jest wersja załadowanego jądra systemu operacyjnego.

Istotna informacja pojawiła się w ostatniej linii. Mówi ona o dacie utworzenia pliku, a więc teoretycznie najstarszym, zapisanym w nim zdarzeniu. W większości systemów plik ten jest automatycznie usuwany co pewien ściśle określony przedział czasu. Czuwa nad tym tablica demona zegarowego *crontab*. Przykłady jej wykorzystania zostały opisane w podrozdziale 4.3. Jeśli natomiast zachodzi konieczność usunięcia, gdyż np. zabrakło miejsca w systemie plików, to w dowolnym momencie czasu może to zrobić użytkownik *root*.

Jeśli komenda *last* zostanie użyta z argumentem będącym nazwą użytkownika w systemie, to wówczas pojawiające się na ekranie opisy zdarzeń będą dotyczyły tylko tego użytkownika. Spośród użytecznych opcji komendy należy wymienić:

- **-t** *rrrrmddhss* – dostarcza informację o historii połączeń do systemu od pierwszego zapamiętanego w pliku dziennika do momentu określonego przez wartość opcji.
- **-d** – powoduje, iż adres hosta, z którego miało miejsce podłączanie do systemu jest podawany jako adres symboliczny.
- **-i** – powoduje, iż adres hosta, z którego miało miejsce podłączanie do systemu jest podawany jako adres IP.
- **-x** – wypisuje dodatkowo informacje o inicjalizacji systemu oraz zmianach poziomu jego pracy. Przejście na poziom 0, to zamknięcie systemu.

Poniżej przedstawiono fragment informacji uzyskanej komendą *last*, uruchomionej z opcją **-x**, ograniczony do czterech zdarzeń. Najstarsze to inicjalizacja i załadowanie jądra systemu operacyjnego, następnie przejście na poziom 3 (normalna, wieloużytkownikowa praca systemu). Kolejne dwa zdarzenia to przejście na poziom 0 oraz zamknięcie systemu. Poziomy pracy systemu zostaną omówione w rozdziale dotyczącym procesów, w tomie drugim.

```

1 [root@messy ~]# last -x
2 shutdown system down 2.6.9-1.667 Fri Mar 6 18:54 - 00:07 (2+05:13)
3 runlevel (to lvl 0) 2.6.9-1.667 Fri Mar 6 18:54 - 18:54 (00:00)
4 .....
5 runlevel (to lvl 3) 2.6.9-1.667 Fri Mar 6 17:22 - 18:54 (01:31)
6 reboot system boot 2.6.9-1.667 Fri Mar 6 17:22 (01:31)

```

### Katalog */var/log*

W katalogu tym przechowywane są dzienniki systemu. Przykładową zawartość przedstawiono poniżej:

```

1 [root@messy log]# ls
2 anaconda.log cups messages-20090126 spooler
3 anaconda.syslog dirmngr messages-20090203 spooler-20090126
4 audit dmesg ntpstats spooler-20090203
5 BackupPC dmesg.old pm-suspend.log tallylog
6 bittorrent faillog ppp tomcat5
7 boot.log gdm prelink vbox
8 boot.log-20090126 httpd rpmpkgs wpa_supplicant.log

```

9	boot.log-20090203	jetty	rpmpkgs-20090126	wtmp
10	btmtp	lastlog	rpmpkgs-20090203	wtmp-20090203
11	btmtp-20090203	mail	samba	Xorg.0.log
12	ConsoleKit	maillog	secure	yum.log
13	cron	maillog-20090126	secure-20090126	
14	cron-20090126	maillog-20090203	secure-20090203	
15	cron-20090203	messages	setroubleshoot	

Jak widać, istotniejsze fragmenty funkcjonalne systemu mają swoje dzienniki. Przykładowo informacje związane z inicjalizacją systemu operacyjnego są zapisywane w pliku *boot.log*. Demon zegarowy, czuwający nad terminowym wykonywaniem zadań w systemie, zapisuje informacje do pliku *cron*. Opis zdarzeń związanych z ogólnie rozumianym bezpieczeństwem systemu trafia do pliku *secure*. W systemach normalnie eksploatowanych liczba generowanych zdarzeń jest bardzo duża. Powoduje to szybki wzrost rozmiaru plików dziennika. W celu panowania nad rozmiarem wprowadzono prosty mechanizm polegający na cyklicznym przenoszeniu zawartości bieżącego pliku dziennika do pliku archiwalnego. Przykładowo, naszym bieżącym plikiem jest plik *secure*, w którym zapisane są zdarzenia od 03.02.2009r. do chwili bieżącej. Plik *rpm\_pkgs-20090203* zawiera zdarzenia, które miały miejsce od 26.01.2009 do 02.02.2009r., zaś plik *rpm\_pkgs-20090126* wcześniejsze. Pliki archiwalne są usuwane, jeśli ich wiek przekroczy zdefiniowany w odpowiedniej tablicy demona zegarowego.

Jak wspomniano, pliki dziennika są plikami tekstowymi. Dla przykładu zobaczmy, jak wyglądała aktywność w systemie użytkownika *jan*. W poniższym fragmencie długie linie zostały podzielone, a ich kontynuacje odsunięte od lewego marginesu. W pliku stanowią one całość.

[illegible]

```

26 Feb  6 19:25:36 messy sshd[3379]: Failed password for jan
27                               from 149.156.99.116 port 33425 ssh2
28 Feb  6 19:25:39 messy sshd[3380]: Disconnecting:
29                               Too many authentication failures for jan
30 Feb  6 19:25:39 messy sshd[3379]: Failed password for jan
31                               from 149.156.99.116 port 33425 ssh2
32 Feb  6 19:25:39 messy sshd[3379]: PAM 2 more authentication failures;
33   logname= uid=0 euid=0 tty=ssh ruser= rhost=thorin.icsr.agh.edu.pl user=jan

```

Wpisy są bardzo szczegółowe. Linia 2–3 zawiera informacje o dadaniu do systemu użytkownika *jan*, zaś linia 4 o stworzeniu w systemie grupy użytkowników o tej samej nazwie. Linia 5–6 informuje o zmianie hasła dla użytkownika. Od tego momentu wprowadzone hasło obowiązuje w systemie. W kolejnych liniach mamy informacje o podłączaniu i odłączaniu użytkownika od systemu. Mamy kilka możliwości podłączania się do systemu. W liniach 7–8 oraz 9–10 znajduje się informacja o poprawnym podłączeniu się w sposób zdalny, przez sieć. Linia 11–12 informuje o podłączeniu się do systemu przez zmianę kontekstu (komenda *su*) z użytkownika *root*. Linia 13–14 to raport o odłączeniu się użytkownika od systemu. Linie 15–17 to informacja o podłączeniu do systemu z konsoli. Istotne, z punktu widzenia bezpieczeństwa systemu są informacje zawarte w liniach 22–33. Do systemu usiłował się podłączyć użytkownik *jan*, ale podawał błędne hasło. Po trzech próbach sesja została przerwana. Informacja pochodzi od systemu PAM. Może świadczyć o próbie włamania przez odgadnięcie hasła. Widać, że prowadzenie w systemie odpowiedniej polityki dotyczącej częstości zmian haseł oraz ich złożoności jest kluczowe. Dotyczy to zwłaszcza użytkownika *root*, którego nazwa jest znana. Stąd często spotyka się systemy, w których zdalne podłączanie dla użytkownika *root* nie jest możliwe. Wówczas podłączamy się jako zwykły użytkownik, a po podłączeniu dokonujemy przełogowania (zmiany kontekstu) komendą *su* na użytkownika *root*.

## 2.2.4 Ograniczanie dostępu do systemu

Na początek ograniczymy się do najprostszych sposobów ograniczenia dostępu do systemu działających na zasadzie „wszystko albo nic”. Następnie omówimy mechanizmy bardziej precyzyjne, udostępnione przez plik *access.defs* znajdujący się w katalogu */etc/security*. Mechanizmy związane z autentykacją zostaną opisane przy okazji opisu systemu PAM w podrozdziale 2.9. Osobnym zagadnieniem będzie kontrola i ograniczanie połączeń zdalnych stanowiące fragment rozdziału 7, dotyczące podstaw konfiguracji sieciowych.

### Metody ogólne

Uniemożliwienie podłączania się do systemu pojedynczemu użytkownikowi, bez konieczności unieważniania hasła polega na zdefiniowaniu odpowiedniego „interpretera poleceń”. W tej rodzinie systemów operacyjnych jest to „interpreter” z katalogu */sbin* o nazwie *nologin*. Wystarczy zatem, będąc administratorem, użyć komendy *chsh* z argumentem będącym nazwą owego użytkownika w systemie i jako nowy interpreter poleceń wstawić */sbin/nologin*. Dokładnie ten sam efekt osiągniemy zmieniając dowolnym edytorem zawartość ostatniego pola linii opisującej użytkownika w pliku */etc/passwd*. Rozwiązanie to działa bardzo prosto. Plik */sbin/nologin* jest najzwyklejszym programem, który wypisuje stosowny komunikat na ekranie i kończy swoje działanie. Każdy potrafi napisać podobny, skompilować i wykorzystywać do podobnych celów. Efekt działania rozwiązania oryginalnego jest następujący:

```

1 [root@messy ~]# su - jan
2 This account is currently not available.
3 [root@messy ~]#

```

Rozwiązanie to działa na każdy sposób podłączania się do systemu, w tym także na zmianę kontekstu z użytkownika *root* komendą *su* na dowolnego innego użytkownika. Działa również na konto użytkownika *root*, co oznacza iż nie będzie on mógł podłączyć się do systemu.

Często zdarza się, że konieczne jest ograniczenie możliwości podłączania się do systemu wszystkim użytkownikom. Sytuacja ta ma miejsce przy okazji dokonywania zmiany wersji systemu, czy instalowaniu oprogramowania. Metoda polegająca na zmianie podstawowego interpretera poleceń z przyczyn praktycznych nie wchodzi w rachubę. Zaproponowane rozwiązanie polega na umieszczeniu w katalogu */etc* pliku o odpowiedniej nazwie. Program logujący sprawdza, czy w katalogu tym znajduje się plik o nazwie *nologin* i jeśli tak, to przerywa proces logowania. Dodatkowo, jeśli plik ten jest plikiem tekstowym, to jego zawartość jest wypisywana na ekranie. Wynik działania tego mechanizmu jest następujący:

```

1 15:15 [bory@thorin ~]$ ssh jan@141.152.99.105
2 jan@141.152.99.105's password:
3
4 *****
5 !   Dzis, 14.02.2009 upgrade systemu           !
6 !   Nie ma mozliwosci podlaczenia sie do systemu   !
7 !   Za utrudnienia przepraszamy                 !
8 *****
9
10 Connection closed by 141.152.99.105

```

Występowanie pliku */etc/nologin* uniemożliwia każdy sposób podłączania się do systemu. Nie działa jedynie na konto użytkownika *root*.

Prace administracyjne wymagające ograniczenia możliwości podłączania się do systemu wszystkim użytkownikom, zwłaszcza przez dłuższy okres czasu są za zwyczaj wcześniej planowane. Stąd, do dobrego tonu należy wcześniejsze poinformowanie o tym użytkowników. Dostępny w systemie mechanizm polega na umieszczeniu w pliku o nazwie *motd* (skrót od: **m**essage **o**f **t**he **d**ay) znajdującym się w katalogu */etc* stosownego komunikatu. W wielu systemach mechanizm ten jest wykorzystywany do informowania użytkowników o zdarzeniach w systemie mających wpływ na jego pracę, co pokazuje poniższy przykład:

```

1 -----
2 University of Minnesota Supercomputing Institute
3 Altix Cluster
4 -----
5 For assistance please contact us at http://www.msi.umn.edu/consult.html,
6 help@msi.umn.edu, or (612)626-0802.
7
8 An introduction to the Altix cluster is available at
9
10 http://www.msi.umn.edu/altix/intro/
11

```

```

12 | To change your password, please use the yppasswd command.
13 | -----
14 | The available scratch spaces on the Altix system are /scratch1, /scratch2
15 | /scratch3 and /scratch4.
16 |
17 | All files in the scratch directories that have not been modified for
18 | 14 days will be deleted. Note that "touch"ing files will not affect
19 | the deletion date. If you need more time, please email help@msi.umn.edu.
20 | -----
21 | Sat Jan 31 18:09:04 CST 2009
22 |
23 | A cooling outage in our datacenter forced us to bring down most of the core
24 | resources, including the Altix. Cooling is now back and the Altix is
25 | online. Re-runnable jobs that were running on the Altix nodes when the
26 | system was shutdown should be re-queued.
27 | -----
28 | Current Scratch File Systems and Quota Information
29 |
30 |      Scratch      Size      Available
31 |      File System   (GB)      (GB)      % Used
32 |      -----
33 |      /scratch1     1142.66     1108.30      4%
34 |      /scratch2     1142.66     1095.33      5%
35 |      /scratch3     1142.66     1125.30      2%
36 |      /scratch4     1142.66     1117.07      3%
37 |
38 | Please use a scratch file system with sufficient free space.
39 | rudolfv@altix [~] %

```

### Plik */etc/security/access.conf*

Plik */etc/security/access.conf* pozwala na kontrolowanie możliwości podłączania się do systemu na poziomie pojedynczego użytkownika lub grupy użytkowników z rozróżnieniem sposobu. Przypomnijmy, że do systemu użytkownik może podłączyć się z konsoli, poprzez zmianę kontekstu (komendą *su*) oraz zdalnie z wykorzystaniem protokołu sieciowego. Plik ten jest plikiem tekstowym. Linia zawiera opis, zwany regułą, pojedynczego ograniczenia podłączania się do systemu. Program logujący sprawdza reguły dostępu zgodnie z zasadą pierwszego dopasowania. Gdy w pliku zostanie znaleziona reguła pasująca do danego połączenia, zostaje natychmiast zastosowana i połączenie zaakceptowane lub odrzucone. Z tego powodu kolejność występowania reguł ma bardzo istotne znaczenie.

W pliku mogą istnieć komentarze. Komentarzem jest fragment linii od pojawienia się znaku *#* do przejścia do nowej linii. W konfiguracji domyślnej nie są zapisane żadne ograniczenia. Plik wprawdzie istnieje, lecz każda jego linia jest komentarzem. Zawartość stanowi opis jego budowy oraz przykłady konfiguracji.

Linia zawierająca regułę logowania składa się z trzech pól oddzielonych znakiem *:*. Kolejne pola oznaczają:

1. W pierwszym polu zapisywany jest symbol dostępu: *+* oznacza, że reguła definiuje prawo dostępu, *-* oznacza, że reguła zawiera definicję blokady dostępu.



2. Drugie zawiera listę nazw użytkowników lub grup, których reguła dotyczy. Składnia dopuszcza użycie słowa kluczowego *ALL* oraz modyfikatora *EXCEPT*, co znacznie upraszcza budowanie reguł. Słowo *ALL* dopasowuje w tym przypadku każdego użytkownika i każdą grupę.
3. W trzecim polu zapisywana jest lista źródeł połączenia. Mogą na niej znaleźć się nazwy hostów, adresy IP hostów w wersji IV lub VI, adresy sieci z podaniem wartości maski po znaku \, nazwy domen oraz słowa kluczowe *LOCAL*, *ALL*, *console* oraz modyfikator *EXCEPT*. Użycie słowa kluczowego *ALL* spowoduje, że dopasowane zostaną wszystkie źródła połączenia. Słowo kluczowe *LOCAL* jest najbardziej skomplikowanym określeniem źródła. Spowoduje ono dopasowanie nazwy hosta nie zawierającej kropki, znajdującego się w tej samej domenie. Mechanizm dopasowania wykorzystuje w tym przypadku odwrotny DNS (ang. *Reverse DNS*). Słowo kluczowe *console* oznacza oprócz podłączania się z konsoli również zmianę kontekstu.

Dla wyjaśnienia kilka przykładów. Poniższa reguła pozwala na podłączanie się do systemu z konsoli jedynie użytkownikowi *root* oraz członkom grupy *bin*:

```
1 - : ALL EXCEPT root bin : console
```

Uniemożliwienie zdalnego podłączania się do systemu wszystkim użytkownikom poza użytkownikiem *root* załatwia następująca reguła:

```
1 - : ALL EXCEPT root : ALL EXCEPT console
```

Reguła:

```
1 + : root : console
```

pozwala na podłączanie do systemu jedynie użytkownikowi *root* wyłącznie z konsoli.

Zapewnienie możliwości podłączania się użytkownika *root* z komputerów o wybranych adresach IP oraz z wykorzystaniem pętli sprzężenia zwrotnego realizują reguły:

```
1 + : root : 192.168.200.1 192.168.200.4 192.168.200.9
2 + : root : 127.0.0.1
```

Aby użytkownik *jan* mógł podłączać się do systemu z komputera pracującego w sieci o zadanym adresie IP w wersji VI i zadanej masce podsieci (w tym przykładzie 64) należy użyć reguły w postaci:

```
1 + : jan : 2001:4ca0:0:101::/64
```

Na koniec przykład ogólnej reguły zabraniającej podłączania się do systemu użytkownikom, którzy nie zostali wyspecyfikowani w poprzedzających ją regułach:

```
1 #- : ALL : ALL
```

### 2.2.5 Usuwanie konta użytkownika

Usunąć użytkownika z systemu może administrator systemu. Najszybszą metodą jest użycie komendy *userdel*. Wymaga ona podania jako argumentu nazwy użytkownika, który ma zostać usunięty. Uruchomiona w ten sposób komenda usunie jedynie linie definiujące użytkownika ze wszystkich plików konfiguracyjnych. Wymagane jest, aby użytkownik nie był aktualnie podłączony do systemu. Zobaczmy, jakie dodatkowe możliwości oferują opcje komendy. Są to:

- **-f** – umożliwia usunięcie użytkownika z systemu nawet jeśli ten jest aktualnie podłączony. Usuwa także cały katalog domowy oraz plik zawierający pocztę użytkownika. Dodatkowo, jeśli w pliku */etc/login.defs* zmienna *USERGROUPS\_ENAB* ma wartość *yes*, to usuwana z systemu jest także, jeśli istnieje, grupa użytkowników o nazwie identycznej z nazwą usuwanego użytkownika nawet jeśli jest ona grupą podstawową dla innego użytkownika.
- **-r** – usuwa zawartość katalogu domowego użytkownika oraz plik przechowujący pocztę użytkownika.

Funkcjonalność komendy *userdel* w dystrybucji Fedora jest relatywnie uboga i w systemach, w których dodawanie i usuwanie użytkowników wykonuje się często lepiej skorzystać z możliwości zdefiniowania własnej. W prosty i elegancki sposób problem usuwania użytkowników z systemu można rozwiązać pisząc skrypt wykorzystujący dostępne w systemie narzędzia. Przykład pokazano i omówiono od strony implementacyjnej w Rozdziale 5.4, w tomie drugim. Jeśli chodzi o jego funkcjonalność, to powinien on realizować:

1. Usunięcie linii definiujących użytkownika z plików konfiguracyjnych - najprostsza realizacja przez wywołanie komendy *userdel* bez opcji, z argumentem będącym nazwą usuwanego użytkownika.
2. Dokonanie archiwizacji katalogu domowego oraz pliku z pocztą użytkownika - wywołanie komendy *tar*. Następnie ich usunięcie.
3. Odnalezienie i usunięcie plików, których właścicielem jest usuwany użytkownik, a zostały one utworzone przez niego bezpośrednio lub w wyniku wykorzystywania dostępnych w systemie usług. Do katalogów, które należy przeglądać należą: */tmp* - z racji przechowywania plików tworzonych przez niektóre aplikacje, */var/spool* - ponieważ mogą znajdować się w nim zadania do uruchomienia przez demon zegarowy, umieszczone komendą *at* oraz tablica zadań do cyklicznego uruchamiania stworzona komendą *crontab*. Przeszukiwanie realizujemy komendą *find*.

Usuwanie z systemu wraz z użytkownikiem plików, których był właścicielem jest zalecane z bardzo prostego powodu, który wyjaśnimy na przykładzie. W katalogu */tmp* znajduje się plik, o nazwie *scisle.tajne*, którego właścicielem jest użytkownik *maciek*.

```
1 -rwxr-xr-x 1 maciek maciek 114296 2009-02-04 14:24 scisle.tajne
```

Opis katalogu domowego użytkownika *maciek* jest następujący:

```
1 drwx----- 6 maciek maciek 4096 2009-02-04 14:30 maciek
```

Użytkownika *maciek* oraz jego grupę podstawową usuwamy z systemu. Od tej pory plik nie ma właściciela ani indywidualnego ani grupowego. W trzeciej i czwartej kolumnie opisu pliku pojawił się numer identyfikacyjny byłego właściciela odpowiednio indywidualnego i grupowego.

```
1 -rwxr-xr-x 1 501 501 114296 2009-02-04 14:24 scisle.tajne
```

Podobnie katalog domowy.

```
1 drwx----- 6 501 501 4096 2009-02-04 14:30 maciek
```

Dodajemy do systemu nowego użytkownika o nazwie *jan*, ale wymuszamy dla niego numer identyfikacyjny użytkownika 501, który jeszcze przed chwilą był wykorzystywany w systemie. Opis pliku *scisle.tajne* z katalogu */tmp* jest teraz następujący:

```
1 -rwxr-xr-x 1 jan jan 114296 2009-02-04 14:24 scisle.tajne
```

Zaś w katalogu */home*, gdzie znajdują się katalogi domowe użytkowników sytuacja jest taka, jak przedstawiono poniżej:

```
1 drwx----- 4 jan jan 4096 2009-02-04 14:52 jan
2 drwx----- 6 jan jan 4096 2009-02-04 14:30 maciek
```

Co się stało i jakie są tego konsekwencje? Otóż jak wspomniano, właściciel jest rozpoznawany po numerze identyfikacyjnym użytkownika. Stąd, jeśli tylko w systemie pojawił się użytkownik, to stał się on właścicielem indywidualnym wszystkich plików, które jako właściciela mają użytkownika o tym numerze. Identycznie sytuacja wygląda z właścicielem grupowym. W konsekwencji, nowy użytkownik ma pełne prawa dostępu do przykładowego pliku w katalogu */tmp* oraz, co gorsza, do katalogu domowego byłego użytkownika *maciek* oraz do pliku przechowującego jego pocztę. Sytuacja jest niezręczna oraz prawnie nieczysta.

## 2.3 Zarządzanie grupami użytkowników w dystrybucji Fedora

### 2.3.1 Dodawanie grupy użytkowników

Grupę użytkowników tworzy w systemie jego administrator. W najprostszy sposób dodanie grupy użytkowników można zrealizować przy pomocy komendy *groupadd*. Komenda ta wymaga podania jedynie nazwy dodawanej grupy. Pozostałe atrybuty zostaną nadane jako domyślne, odczytane wprost z plików konfiguracyjnych lub obliczone na podstawie ich zawartości. Z częściej stosowanych opcji komendy należy wymienić:

- **-g numer\_identyfikacyjny\_grupy** – pozwala na nadanie grupie numeru identyfikacyjnego, który został podany jako wartość opcji.
- **-p hasło** – umożliwia nadanie hasła grupowego. Hasło musi zostać podane w postaci zaszyfrowanej, w formacie: *\$identyfikator\$sól\$hasło*.
- **-o** – pozwala na utworzenie grupy z numerem identyfikacyjnym, który posiada już istniejąca grupa w systemie (nieunikalnym).

W praktyce utworzenie nowej grupy w systemie przebiega następująco:

```
1 [root@messy ~]# groupadd -g 700 studenci
```

Wykonanie komendy spowodowało, że w plikach konfiguracyjnych grup użytkowników pojawiły się następujące wpisy:

- w pliku */etc/group*:

```
1 studenci:x:700:
```

- w pliku */etc/gshadow*:

```
1 studenci:::
```

Dodana w ten sposób grupa nie posiada żadnych członków, czyli nie jest podstawową ani dodatkową dla żadnego użytkownika. Nie ma ustawionego hasła oraz nie posiada administratora. Przeglądając plik */etc/passwd* stwierdzimy również, że nie jest ona grupą podstawową dla żadnego użytkownika.

### 2.3.2 Zmiana konfiguracji grupy

Czynności administracyjne dotyczące grupy mogą być dwojakiego rodzaju. Po pierwsze mogą dotyczyć zmian wartości podstawowych atrybutów grupy. Te najczęściej dokonujemy z wykorzystaniem komendy *groupmod*. Drugi rodzaj, to czynności związane z administrowaniem użytkownikami. Chodzi tu o zmianę przynależności użytkowników do grup, zarówno podstawowych jak i dodatkowych. Tu również z pomocą przychodzi komenda *groupmode*. Administrator systemu może również ustalić lub zmienić administratora dla danej grupy. Administrator grupy, jak wspomniano, może dodać użytkownika do grupy lub go z niej usunąć. Aby jednak jego poczynania były kontrolowane, wprowadzono hasło grupowe, które ustala administrator systemu. Czynności te najprościej wykonać komendą *gpasswd*.

Zacznijmy od zmiany podstawowych atrybutów grupy. Komendę *groupmod* wywołujemy zawsze z argumentem będącym nazwą grupy w systemie, wartości atrybutów której poddajemy modyfikacjom. Najczęściej wykorzystywane opcje komendy to:

- **-g** *numer\_grupy* – pozwala zmienić numer grupy na podany jako wartość opcji. Jeśli zaproponowany numer jest już przypisany innej grupie, to wykonanie komendy kończy się niepowodzeniem.
- **-n** *nazwa\_grupy* – umożliwia zmianę nazwy grupy na podaną jako wartość opcji. Podobnie, jak w przypadku opcji **-g**, jeśli grupa o zaproponowanej nazwie w systemie już istnieje, nazwa grupy nie zostanie zmieniona.
- **-p** *postać\_zaszyfrowana\_hasła* – służy do ustawienia hasła grupowego. Ponieważ wymagane jest podanie hasła w postaci *\$identyfikator\$sól\$hasło*, opcje tą wykorzystuje się rzadko.

Dla przykładu zmienimy nazwę zdefiniowanej uprzednio grupy *studenci* na *pracownicy* oraz jej numer na 800. Komenda ma postać:

```
1 [root@messy ~]# groupmod -n pracownicy -g 800 studenci
```

W chwili obecnej grupa ta nie posiada żadnego użytkownika. Niech stanie się ona podstawową dla użytkownika *antek*:

```
1 [root@messy ~]# usermod -g pracownicy antek
```

Linia opisująca użytkownika *antek* w pliku */etc/passwd* wygląda teraz następująco:

```
1 antek:x:502:800::/home/antek:/bin/bash
```

Będąc administratorem systemu przekażemy uprawnienia do zarządzania grupą *pracownicy* użytkownikowi *antek*. Najprościej można to zrealizować przy pomocy komendy *gpasswd*. Wymaga ona podania jako argumentu nazwy grupy, dla której będą przeprowadzane zmiany. Jeśli nie podamy opcji, to wykonanie komendy będzie polegało na zmianie hasła grupowego. Spośród opcji komendy, najczęściej stosowane to:

- **-a** *nazwa\_użytkownika* – pozwala uczynić grupę dodatkową dla użytkownika, którego nazwa w systemie pojawiła się jako wartość opcji.
- **-d** *nazwa\_użytkownika* – usuwa użytkownika o podanej nazwie z grupy.
- **-r** – umożliwia usunięcie hasła grupowego.
- **-A** *nazwa\_użytkownika,...* – pozwala zdefiniować administratorów grupy poprzez podanie listy ich nazw w systemie oddzielonych przecinkiem (w liście nie może pojawić się żaden biały znak).
- **-M** *nazwa\_użytkownika,...* – podobnie jak opcja **-a**, umożliwia dodanie użytkowników, których nazwy w systemie oddzielone przecinkami pojawiły się jako wartość opcji, do grupy.

Kolejny krok w zarządzaniu grupą polega zatem na wykonaniu komendy:

```
1 [root@messy ~]# gpasswd -A antek pracownicy
```

Wynik jej działania odnajdziemy w pliku */etc/gshadow*, gdzie w trzeciej kolumnie linii opisującej grupę *pracownicy* pojawił się identyfikator użytkownika, jako administratora grupy:

```
1 pracownicy:!:antek:
```

Ustalmy jeszcze hasło grupowe:

```
1 [root@messy ~]# gpasswd pracownicy
2 Changing the password for group pracownicy
3 New Password:
4 Re-enter new password:
```

Zostało ono zapisane w drugiej kolumnie pliku */etc/gshadow*:

```
1 pracownicy:$1$.HIZsP9.$Uqophy9.ARKFUDGyRRN5M1:antek:
```

Użytkownik będący administratorem grupy może dodać użytkownika do grupy:

```
1 [antek@messy ~]$ gpasswd -a jan pracownicy
2 Adding user jan to group pracownicy
```

### 2.3.3 Usuwanie grupy użytkowników

Usuwanie grupy użytkowników z systemu jest czynnością wykonywaną znacznie rzadziej niż dodawanie grupy. Co pewien czas, zwłaszcza po wykonaniu operacji usunięcia użytkownika z grupy lub przeniesienia go do innej grupy należy sprawdzić, czy czynności te nie spowodowały, że w systemie pojawiła się grupa, do której nie należy żaden użytkownik. Przypomnijmy, że informacji o tym, czy dana grupa jest podstawową dla jakiegoś użytkownika systemu szukamy w pliku */etc/passwd*. Jeśli numeru interesującej nas grupy nie znajdziemy w czwartej kolumnie żadnej linii pliku, to nie jest ona podstawową dla żadnego użytkownika. Lista członków grupy znajduje się w pliku */etc/group*. Jeśli czwarta kolumna linii definiującej daną grupę w tym pliku jest pusta, to grupa ta nie jest dodatkową dla żadnego użytkownika. W takim przypadku grupa może zostać usunięta.

Grupę użytkowników może z systemu usunąć użytkownik *root*. Ma on do dyspozycji komendę *groupdel*, którą wywołuje się z argumentem będącym nazwą grupy w systemie, która ma zostać usunięta. Działanie komendy sprowadza się jedynie do usunięcia linii definiujących grupy z plików */etc/group* oraz */etc/gshadow*. Dodatkowo, przed usunięciem, sprawdzane jest, czy grupa nie jest podstawową dla jakiegoś użytkownika w systemie. Usunięcie takiej grupy spowodowałoby problemy z określeniem właściciela grupowego w momencie tworzenia przez użytkownika nowego pliku. Przy tej okazji należy zwrócić uwagę na fakt, iż usuwając grupę komenda *userdel* nie sprawdza, czy w systemie plików nie zostały pliki, których właścicielem grupowym jest usuwana grupa. Dla takich plików najprościej zmienić właściciela grupowego na grupę podstawową właściciela indywidualnego. Reasumując, podobnie jak dla problemu usuwania z systemu użytkownika, najlepszym rozwiązaniem będzie napisanie skryptu, które zrealizuje dwie czynności:

1. Znajdzie w systemie pliki, których właścicielem grupowym jest usuwana grupa i dokona jego zmiany (komenda *chown*).
2. Usunie definicje grupy z plików konfiguracyjnych komendą *groupdel*.

Działanie komendy *groupdel* w przypadku, gdy usiłujemy usunąć grupę, która jest podstawową dla co najmniej jednego użytkownika jest następujące:

```
1 [root@messy ~]# groupdel pracownicy
2 groupdel: cannot remove user's primary group.
3 [root@messy ~]#
```

Jeśli natomiast grupa nie jest podstawową dla żadnego użytkownika, lub wogóle nie posiada członków, usunięcie jej z systemu przebiega bez żadnego komunikatu:

```
1 [root@messy ~]# groupdel antek
2 [root@messy ~]#
```

## 2.4 Spójność plików konfiguracyjnych w dystrybucji Fedora

Po lekturze podrozdziałów opisujących budowę plików konfiguracyjnych dla użytkowników i grup oraz analizie działania mechanizmów dodawania, zmiany atrybutów i usuwania użytkownika oraz grupy można dojść do wniosku, że ten zakres czynności administratora można z powodzeniem zrealizować posługując się edytorem tekstowym na odpowiednich plikach konfiguracyjnych. Wniosek ten jest jak najbardziej słuszny pod warunkiem posiadania przez administratora odpowiedniego

doświadczenia. Należy mieć świadomość, że rezygnując w wykorzystywaniu odpowiednich komend narażamy się dodatkowo na błędy w danych zapisywanych do plików. Komendy posiadają dość dobre mechanizmy testowania poprawności danych. Przykładowo, komenda *useradd* nie pozwoli na dodanie użytkownika, którego nazwa składa się z samych cyfr, zawiera znaki specjalne (np. ":") zaś proponowany numer identyfikacyjny jest już w systemie wykorzystywany przez innego użytkownika. Dodatkowo, w przypadku użytkowników systemu oraz grup użytkowników informacja o nich zapisana jest w kilku plikach. Dlatego doprowadzenie do sytuacji, w której informacja przestaje być spójna jest praktycznie bardzo łatwe. Prosty przykład to usunięcie z systemu grupy będącej podstawową dla użytkownika. Z plików */etc/group* oraz */etc/gshadow* usuwamy edytorem po jednej linii. W pliku */etc/passwd* pozostaje numer grupy, która faktycznie w systemie nie istnieje. Stąd powstały programy służące do kontroli poprawności składniowej oraz spójności informacji zapisanej w plikach konfiguracyjnych użytkowników i grup. Występują w każdej dystrybucji systemu Unix, choć różnią się nazwami i sposobem działania. W dystrybucji Fedora są to *pwck* oraz *grpck*.

### 2.4.1 Pliki konfiguracyjne użytkowników

Komenda *pwck* służy do badania informacji związanej z autentykacją użytkownika w systemie. Przedmiotem jej zainteresowania są pliki *passwd* oraz *shadow*. Jeśli komenda zostanie wywołana bez argumentów, to pracuje ona na plikach systemowych z katalogu */etc*. Stąd zalecane jest, zwłaszcza dla mniej doświadczonych administratorów, skopiowanie obu plików np. do katalogu */tmp* oraz podanie jako argumentów bezwzględnych ścieżek dostępu do obu plików w kolejności *passwd shadow*. Wówczas sprawdzimy błędy w plikach konfiguracyjnych, a jeśli komenda dokona zmian, to nie wpłyną one na pracę systemu. Następnie możemy przystąpić do modyfikowania plików systemowych. Komenda posiada trzy opcje:

1. **-q** – raportuje tylko błędy, pomijając ostrzeżenia.
2. **-r** – działa w trybie *tylko do odczytu*, co powoduje, że żadne poprawki proponowane przez komendę nie będą wprowadzane.
3. **-s** – sortuje informacje w plikach *passwd* oraz *shadow* według numerów identyfikacyjnych użytkowników.

W trakcie działania programu sprawdzane jest:

- czy w każdej linii występuje poprawna liczba pól,
- czy nazwa użytkownika nie powtarza się,
- czy numer identyfikacyjny użytkownika i grupy podstawowej jest poprawny,
- czy grupa podstawowa użytkownika jest poprawnie zdefiniowana w systemie,
- czy ścieżka dostępu do katalogu domowego użytkownika jest poprawna (katalog istnieje),
- czy ścieżka dostępu do podstawowego interpretera poleceń jest poprawna (plik istnieje).

Jako błąd krytyczny komenda traktuje niepoprawną liczbę pól oraz powtórzoną nazwę użytkownika. Linie, w których znaleziony zostanie jeden z tych błędów mogą zostać usunięte po uprzednim wyrażeniu zgody przez administratora systemu, który jest uprawniony do wykonywania komendy *pwck*. Pozostałe braki traktowane są jak ostrzeżenia. Informacja o ich występowaniu jest jedynie raportowana. Komenda, po zakończeniu działania zwraca do procesu macierzystego

(podstawy teorii procesów przedstawiono w Rozdziale 4) informację o tym jak przebiegło jej wykonanie. Informacja ta jest w postaci liczby bajtowej i jest nazywana kodem powrotu. W przypadku komendy *pwck* mamy następujące możliwości:

- **0** – wykonanie zakończyło się sukcesem. Nie znaleziono żadnych błędów.
- **1** – pojawił się błąd w składni wywołania komendy (np. niepoprawna kolejność plików lub podanie niewystępującej opcji).
- **2** – znaleziono jeden lub więcej błędów w plikach konfiguracyjnych użytkowników.
- **3** – próba otwarcia plików konfiguracyjnych zakończyła się niepowodzeniem. Może to być wynikiem podania błędnej ścieżki dostępu lub zabronieniem dostępu (lock) przez inny proces.
- **4** – próba zabronienia dostępu do plików konfiguracyjnych zakończyła się niepowodzeniem. Sytuacja taka ma najczęściej miejsce, gdy inny proces pracuje na plikach konfiguracyjnych i wcześniej wykonał operację zabronienia dostępu.
- **5** – próba wprowadzenia zmian do plików konfiguracyjnych nie powiodła się.

Prześledźmy działanie komendy *pwck* w praktyce. W tym celu pliki */etc/passwd* oraz */etc/shadow* skopiowano do katalogu */tmp*. Do pliku *passwd* wprowadzono 3 błędy: w linii definiującej użytkownika *jan* usunięto pole siódme definiujące podstawowy interpreter poleceń, skopiowano linię definiującą użytkownika *antek* oraz użytkownikowi *root* zdefiniowano nieistniejący interpreter poleceń. Wynik działania komendy jest następujący:

```

1 [root@messy tmp]# pwck /tmp/passwd /tmp/shadow
2 user root: program /bin/ksh does not exist
3 user adm: directory /var/adm does not exist
4 user uucp: directory /var/spool/uucp does not exist
5 user gopher: directory /var/gopher does not exist
6 user ftp: directory /var/ftp does not exist
7 duplicate password entry
8 delete line 'antek:x:502:800::/home/antek:/bin/bash'? y
9 user jan: directory Jan Kowalski,C2 pok.433,6179999,+0122345678 does not exist
10 invalid password file entry
11 delete line 'jan:x:501:501:Jan Kowalski,C2 pok.433,6179999,45678:/home/jan'? y
12 no matching password file entry in /tmp/passwd
13 delete line 'jan:$6$kvAtcxpo$YBz8ERxveATu4yp8ujR3Y5U4epqWWRz7YggrZobpaJJD4
14     TUVdzcWv4yMv5UrYCPQWxTwrhkG0rQ99dCJ/t8Es/:14279:0:99999:7:::'? y
15 pwck: the files have been updated
16 [root@messy tmp]# echo $?
17 2
18 [root@messy tmp]#

```

W linii 2, jako ostrzeżenie, pojawił się komunikat o niepoprawnej ścieżce dostępu do interpretera poleceń użytkownika *root*. Linie 3–6 zawierają informację o tym, że w systemie plików nie istnieją katalogi domowe niektórych użytkowników. Tym nie przejmujemy się, gdyż konta tych użytkowników służą do propagacji praw własności, a nie do podłączania się do systemu. Linie



7–8 to reakcja na powtarzający się identyfikator użytkownika będący wynikiem skopiowania linii definiującej użytkownika *antek*. Linia 8 zawiera pytanie, czy usunąć linię z powtarzającym się identyfikatorem. Linie 9–14 stanowią reakcję na brakujące pole z definicją podstawowego interpretera poleceń. Po usunięciu wadliwej linii z pliku *passwd* (linia 11), plik *shadow* zawiera „nadmiarową” linię. Stąd propozycja jej usunięcia (linie 13–14). Linia 15 zawiera informację, że pliki konfiguracyjne zostały zmodyfikowane. Następnie sprawdzamy, jaką wartość zwróciła ostatnio wykonana komenda (linie 16, 17). Otrzymujemy wartość 2, co oznacza, że w plikach konfiguracyjnych użytkowników występowały błędy.

### 2.4.2 Pliki konfiguracyjne grup

Dla sprawdzania poprawności plików definiujących grupy użytkowników służy komenda *grpck*. Komenda pracuje na plikach *group* oraz *gshadow*. Komenda wywołana bez argumentów pracuje na plikach systemowych z katalogu */etc*. Stąd, podobnie jak w przypadku badania plików konfiguracyjnych użytkowników, zalecane jest skopiowanie plików systemowych np. do katalogu */tmp*, sprawdzenie ewentualnych błędów na kopiach, a następnie powtórzenie czynności na plikach oryginalnych. Jeśli decydujemy się na pracę na plikach oryginalnych, to pierwsze uruchomienie komendy najlepiej przeprowadzić z opcją zapewniającą, iż żadne zmiany w plikach konfiguracyjnych nie zostaną wprowadzone.

Komenda *grpck* posiada 2 opcje. Są to:

- **-r** – zapewnia pracę bez wprowadzania poprawek w plikach konfiguracyjnych (tryb pracy tylko do odczytu).
- **-s** – wpisy w plikach konfiguracyjnych są analizowane według rosnącego numeru identyfikacyjnego grupy, a nie w kolejności występowania w pliku.

Każda linia definiująca grupę w plikach konfiguracyjnych sprawdzana jest pod kątem:

- poprawnej liczby pól,
- unikalności nazwy grupy,
- poprawności listy członków i administratorów.

Dwa pierwsze błędy traktowane są jako krytyczne i jeśli komenda nie została uruchomiona z opcją **-r**, to jej reakcją będzie pytanie o usunięcie błędnej linii. Trzeci błąd jest traktowany jako ostrzeżenie i jedynym efektem jego wystąpienia jest wypisanie komunikatu na ekranie. Informację o przebiegu wykonania komendy można odczytać na podstawie wartości zwróconej do procesu macierzystego. Są one identyczne, jak w przypadku komendy *pwck*.

Dla pokazania działania komendy *grpck*, pliki */etc/group* oraz */etc/gshadow* skopiowano do katalogu */tmp* oraz wprowadzono w pliku *group* dwa błędy. Pierwszy polegał na dodaniu w linii definiującej grupę *pracownicy* pustego, piątego pola. Drugi na skopiowaniu linii definiującej grupę *root* w pliku *gshadow*. Wynik działania jest następujący:

```
1 [root@messy tmp]# grpck /tmp/group /tmp/gshadow
2 duplicate shadow group entry
3 delete line 'root:::root'? y
4 grpck: the files have been updated
5 [root@messy tmp]# echo $?
6 2
7 [root@messy tmp]#
```

Jak widać komenda nie zareagowała na pole nadmiarowe dodane na końcu linii. Jedynym błędem okazała się powtórzona nazwa grupy. Stąd w liniach 2–3 pytanie o ewentualne jej usunięcie. Na końcu informacja o zmodyfikowaniu plików, na których komenda pracowała. Komenda zwróciła wartość 2 informującą o występujących błędach w plikach konfiguracyjnych grup.

## 2.5 Podstawowe pliki konfiguracyjne w dystrybucji FreeBSD

W systemie FreeBSD pliki konfiguracyjne użytkowników i grup są plikami tekstowymi o budowie linikowej. Linia opisująca użytkownika lub grupę jest podzielona na pola przy pomocy znaku specjalnego, jakim jest ":". Dodatkowo dla przyspieszenia pracy systemu wprowadzono bazę danych użytkowników.

### 2.5.1 Plik */etc/master.passwd*

Fragment pliku z dystrybucji FreeBSD przedstawiono poniżej.

```

1 root:$1$qoRiT0n0$t/1neYLk4DASrLgrKRSS.0:0:0::0:0:Charlie &:/root:/bin/csh
2 toor:*:0:0::0:0:Bourne-again Superuser:/root:
3 daemon*:1:1::0:0:Owner of many system processes:/root:/usr/sbin/nologin
4 operator*:2:5::0:0:System &:/usr/sbin/nologin
5 .....
6 polkit*:562:562::0:0:PolicyKit Daemon User:/nonexistent:/sbin/nologin
7 wacek:$1$UjAvonVC$1cY8vh1CMJD/zGk/cbEcm1:1001:1001::1235865600:0:Waclaw Kowalski:/home/wacek:

```

Jak wspomniano, plik ten jest podstawowym plikiem konfiguracyjnym użytkowników w rodzinie systemów operacyjnych BSD. Na podstawie jego zawartości może zostać utworzony plik */etc/passwd* oraz zmodyfikowana baza danych użytkowników systemu. Podobnie, jak inne pliki konfiguracyjne jest plikiem tekstowym o budowie linikowej. Linie, w których pierwszym nie białym znakiem jest znak "#" są traktowane jako kometarz (ignorowane). Jedna linijka zawiera konfigurację jednego użytkownika. Jest podzielona na 10 pól oddzielonych separatorem w postaci znaku ":". Pola przechowują wartości następujących atrybutów opisujących użytkownika:

1. Pole pierwsze zawiera nazwę użytkownika w systemie.
2. W drugim polu przechowywana jest hasło użytkownika w postaci zaszyfrowanej. Jego format jest taki sam, jaki występuje w drugiej kolumnie pliku */etc/shadow*. Ogólna postać, to *\$identyfikator\$sól\$hasło*, gdzie *identyfikator* oznacza identyfikator algorytmu szyfrującego (patrz tabela 2, *sól* przypadkowy ciąg znaków dodany do szyfrowanego hasła, zaś *hasło* zaszyfrowane hasło użytkownika. W tej rodzinie systemów, do blokowania konta wykorzystuje się znak "\*". Jeśli chcemy zablokować hasło, to przed jego postacią zaszyfrowaną wstawiamy również znak "\*".
3. Pole trzecie zawiera numer identyfikacyjny użytkownika w systemie.
4. Czwarte pole przechowuje numer identyfikacyjny tzw. grupy logowania. Jest to po prostu grupa podstawowa użytkownika.
5. W piątym polu przechowywana jest informacja o klasie logowania użytkownika. Klasa logowania określa ilość zasobów systemowych, które będą dostępne użytkownikom do niej

należącym. Do zasobów tych można zaliczyć czas procesora, liczbę otwartych plików, maksymalny rozmiar pliku, dostępny obszar pamięci operacyjnej, itd. Definicje grup logowania oraz limity dostępnych dla nich zasobów znajdują się w pliku */etc/login.conf*.

6. Pole szóste przechowuje informacje o dacie ważności hasła w postaci liczby sekund, które upłynęły od 1 stycznia 1970 roku do momentu, w którym system wymusi zmianę hasła. Wartość 0 oznacza, iż moment zmiany hasła nie został wyznaczony. Wartość tego pola można ustalać komendami *chpass* lub *pw*. Komenda ta dokonuje automatycznego przeliczenia formatu *dzień miesiąc rok* wymaganą liczbę sekund. Ograniczenie to ma charakter jednorazowy. Po zmianie hasła w polu tym wpisywana jest wartość 0.
7. Wartość pola szóstego jest wykorzystywana do zakładania kont okresowych, blokowanych automatycznie po upływie określonego czasu. Określa ona liczbę sekund, które mają upłynąć od 1 stycznia 1970 roku do momentu automatycznego zablokowania konta. Wartość 0 oznacza, iż ograniczenie to nie zostało wprowadzone. Zadana wartość można wprowadzić komendą *chpass*.
8. Pole ósme przechowuje informacje o użytkowniku w postaci identycznej, jak w piątym polu linii definiującej użytkownika pliku */etc/passwd*.
9. W polu dziewiątym przechowywana jest ścieżka dostępu do katalogu domowego użytkownika.
10. Pole dziesiąte zawiera ścieżkę dostępu do podstawowego interpretera poleceń użytkownika.

Prawa dostępu do pliku */etc/master.passwd* zostały tak ustawione, że umożliwiają jego edycję jedynie właścicielowi indywidualnemu, czyli użytkownikowi *root*. W dokumentacji systemu możemy znaleźć sugestie, aby plik ten edytować nie przy pomocy zwykłego edytora, a przy pomocy komendy *vipw*. Komenda ta pozwala na edycję pliku, domyślnie edytorem *vi*, a po jej zakończeniu sprawdza poprawność składni pliku oraz zawartej w nim informacji. W tym zakresie komenda ta pełni zatem rolę podobną, jak komenda *pwck* w dystrybucji Fedora, której działanie omówiono dokładnie w punkcie 2.8

Komenda *pwd\_mkdb*, tworzy plik */etc/passwd* usuwając z pliku *master.passwd* informację o klasie logowania użytkownika oraz pola z datami ważności hasła i konta. Postać zaszyfrowaną hasła zastępuje znakiem *"*". Dokonuje również modyfikacji pliku */etc/pwd.db* będącego plikiem bazy danych użytkowników systemu. Komendę można uruchomić jawnie z linii komend. Niektóre programy narzędziowe wywołują ją na zakończenie swojej pracy, w celu uaktualnienia zawartości pozostałych plików konfiguracyjnych.

### 2.5.2 Plik */etc/passwd*

Plik przechowujący podstawowe informacje o zdefiniowanych w systemie użytkownikach nosi nazwę *passwd* i znajduje się w katalogu */etc*. Jego budowa jest identyczna, jak opisana w punkcie 2.1.1 budowa pliku */etc/passwd* w dystrybucji Fedora. Dopuszcza się istnienie linii komentujących, ignorowanych przez aplikacje pracujące na pliku. W liniach tych pierwszym, nie białym znakiem jest *"#"*.

Prawa dostępu umożliwiają odczyt pliku każdemu użytkownikowi systemu, jego modyfikacja jest możliwa jedynie dla użytkownika *root*. Jak wspomniano, w systemach rodziny BSD plik */etc/passwd* można wygenerować z pliku *master.passwd*, uważanego za podstawowy plik konfiguracyjny użytkowników. Służy do tego komenda *pwd\_mkdb*.

### 2.5.3 Plik */etc/pwd.db*

Plik */etc/pwd.db* jest bazą danych, której zawartość odpowiada informacji zapisanej w plikach */etc/passwd* oraz */etc/master.passwd*. Zastosowany format umożliwia szybki odczyt wartości atrybutów użytkownika w przypadku dużej liczby użytkowników zdefiniowanych w systemie (inne systemy stosują indeksowanie pliku */etc/passwd*). Zawartość pliku */etc/pwd.db* może odczytać każdy użytkownik systemu. Nie stanowi to żadnego zagrożenia, gdyż w bazie danych przechowywana jest informacja z pliku m.in. */etc/passwd*, która jak wiadomo jest jawna.

Wprowadzenie kolejnego pliku przechowującego informacje o użytkownikach systemu, powoduje podniesienie poziomu dbałości o spójność informacji w nich zapisanych. Stąd zalecane jest wykorzystywanie programów narzędziowych typu *passwd* czy *vipw* zamiast zwykłego edytora do pracy na tekstowych plikach */etc/passwd* oraz */etc/master.passwd*. Na zakończenie pracy programów *passwd* oraz *vipw* wywoływany jest program *pwd\_mkdb*, który modyfikuje zawartość pliku */etc/pwd.db*. Konsekwencje braku spójności plików konfiguracyjnych użytkowników są trudne do przewidzenia.

### 2.5.4 Plik */etc/group*

Plik zawierający definicję grup występujących w systemie znajduje się w katalogu */etc* i nazywa się *passwd*. Jest on plikiem tekstowym o budowie linikowej. Jedna linijka zawiera definicję jednej grupy. Podzielona jest znakiem `:` na cztery pola. Linie, w których pierwszym nie białym znakiem jest znak `#` są traktowane jako komentarz. W linii nie powinno znajdować się więcej niż 1024 znaki. Jeśli linia jest dłuższa, to znaki od 1025 są ignorowane. Fragment pliku pochodzący z systemu FreeBSD zamieszczono poniżej.

```

1 # $FreeBSD: src/etc/group,v 1.35 2007/06/11 18:36:39 ceri Exp $
2 #
3 wheel:*:0:root,wacek
4 daemon:*:1:
5 kmem:*:2:
6 .....
7 polkit:*:562:
8 wacek:*:1001:

```

Kolejne pola linii oznaczają:

1. W pierwszym polu znajduje się nazwa grupy. Powinna być ona unikalna w systemie.
2. Drugie pole ma znaczenie historyczne. Dawniej znajdowała się w nim postać zakodowana hasła grupowego. Ponieważ zawartość pliku, w tym zakodowane hasło, może zostać odczytane przez każdego użytkownika systemu, zrezygnowano z przechowywania go w tym pliku. Dla zachowania spójności z wcześniejszymi wersjami pole pozostawiono, lecz znajduje się w nim `*`. Ma to oznaczać, iż hasło jest zablokowane. W tej rodzinie systemów Unix zrezygnowano z wykorzystywania hasła grupowego, podając jako powód małą jego skuteczność w podnoszeniu poziomu bezpieczeństwa systemu.
3. Pole trzecie przechowuje numer identyfikacyjny grupy (GID). Podobnie jak nazwa grupy powinien on być unikalny w systemie.
4. W ostatnim polu przechowywana jest lista nazw użytkowników, dla których grupa ta jest grupą dodatkową. W systemie ograniczono liczbę członków grupy do 255 użytkowników.

Również w tej wersji systemu, zdefiniowane zostały grupy, które służą do przeniesienia wybranych aspektów administrowania systemem na zwykłych użytkowników systemu. Odbywa się to dzięki wykorzystaniu praw dostępu do plików zawierających komendy systemowe, określonych dla właściciela grupowego. Jeżeli tylko dana grupa, będąca właścicielem grupowym pliku ma prawo do wykonywania go, to każdy użytkownik należący do niej ma takie prawo. Reasumując, o możliwościach użytkownika w systemie decyduje to, do jakich grup należy. Podstawowe grupy predefiniowane to:

- **bin** – grupa dla programów ogólnego przeznaczenia.
- **wheel** – grupa użytkowników, która jest upoważniona do korzystania z konta użytkownika *root*. W praktyce umożliwia podłączenie się do systemu z konta zwykłego użytkownika na konto użytkownika *root* przez zmianę kontekstu komendą *su*.
- **kmem** – grupa wykorzystywana przez programy, które mają dostęp do pamięci, jak np. *netstat* czy *fstat*.
- **mail** – grupa wykorzystywana przez programy obsługujące pocztę elektroniczną.
- **daemon** – grupa dla demonów usług sieciowych. Obsługuje np. kolejki drukarkowe.
- **staff** – grupa dla administratorów systemu.
- **nogroup** – grupa nie posiadająca żadnych uprawnień w systemie.

## 2.6 Zarządzanie użytkownikami w systemie FreeBSD

Podstawowymi komendami, służącymi do zarządzania użytkownikami w systemie FreeBSD są:

- *adduser* - dodawanie użytkownika do systemu,
- *passwd* - zarządzanie hasłami użytkowników,
- *chpass* - zmiana atrybutów użytkownika,
- *rmuser* - usuwanie użytkownika z systemu,
- *pw* - ogólne narzędzie do modyfikowania atrybutów użytkowników i grup w systemie (nazywane edytorem użytkowników).
- *chfn* - zmiana opisu użytkownika (zwartości odpowiedniej kolumny w pliku */etc/master.passwd*),
- *chsh* - zmiana podstawowego interpretera użytkownika.

### 2.6.1 Dodawanie użytkownika do systemu

Omówienie metodyki dodawania użytkownika do systemu rozpoczniemy od podania ograniczeń, które wprowadzono w systemie na wartości atrybutów opisujących użytkownika oraz ogólnych reguł, które należy przestrzegać podczas tego procesu. W dokumentacji zostało powiedziane, że:

- **Nazwa użytkownika w systemie** powinna składać się z małych liter, cyfr i niektórych znaków specjalnych. Nie może rozpoczynać się znakiem "-", jak również składać się z samych cyfr. Maksymalna długość nazwy to 16 znaków.

- **Opis użytkownika** w systemie może składać się z czterech części oddzielonych przecinkiem. Kolejne pola są traktowane przez aplikacje jako: imię i nazwisko użytkownika, numer jego pokoju, telefon służbowy oraz telefon domowy. W opisie oczywiście nie może pojawić się znak ":".
- **Podstawowym interpreterem poleceń użytkownika** może być interpreter, który został zdefiniowany w pliku `/etc/shells`. Jako wartość opcji podajemy bezwzględną ścieżkę dostępu do wybranego interpretera poleceń lub jego nazwę (`/bin/bash` lub tylko `bash`).
- **Numer identyfikacyjny użytkownika** może zostać ustalony przez administratora lub wygenerowany. Generacja odbywa się na zasadzie poprzednio wykorzystany powiększony o jeden, przy czym komenda sama sprawdza jego unikalność i wypełnia powstałe luki w numeracji. Wartość nie może być większa niż 32000, zaś najmniejsza możliwa do przypisania użytkownikowi została w tym systemie ustalona na 1001.
- **Numer grupy podstawowej użytkownika** Podobnie jak numer identyfikacyjny użytkownika, może zostać zaproponowany lub wygenerowany z zachowaniem unikalności. Maksymalna wartość nie może być większa niż 32000. System rozpoczyna numerację nowo dodawanych grup od wartości 1001.
- **Hasło** może zostać zaproponowane jako puste, zdefiniowane przez administratora w momencie dodawania użytkownika do systemu lub ustalone przez użytkownika. Może zostać wygenerowane jako ciąg przypadkowych znaków lub zostać zablokowane.

W praktyce definiowanie nowego użytkownika w systemie można przeprowadzić „ręcznie”, dodając odpowiednie wpisy w plikach konfiguracyjnych oraz tworząc katalog domowy użytkownika i kopiując do niego pliki konfiguracyjne dla interpreterów poleceń lub innych aplikacji. Należy pamiętać o korzystaniu z programów narzędziowych, które od razu sprawdzą poprawność wprowadzonych danych oraz uaktualnią zawartość bazy danych użytkowników. Aby użytkownik mógł podłączyć się do systemu, należy jeszcze wprowadzić hasło. Alternatywą jest skorzystanie z komendy `adduser`. Komenda ta jest skryptem wykorzystującym komendę `pw`. Zasadniczo pracuje ona w sposób interaktywny, ale istnieje możliwość podania wartości atrybutów użytkownika z pliku tekstowego o odpowiednim formacie. Plikiem konfiguracyjnym komendy `adduser` jest plik `/etc/adduser.conf`. Po zainstalowaniu systemu, plik ten nie istnieje i komenda korzysta z wartości domyślnych. W takim wypadku dodanie użytkownika ma następujący przebieg:

```

1 # adduser
2 Username: jan
3 Full name: Jan Kowalski
4 Uid (Leave empty for default):
5 Login group [jan]:
6 Login group is jan. Invite jan into other groups? []:
7 Login class [default]:
8 Shell (sh csh tcsh nologin) [sh]:
9 Home directory [/home/jan]:
10 Use password-based authentication? [yes]:
11 Use an empty password? (yes/no) [no]:
12 Use a random password? (yes/no) [no]:
13 Enter password:
14 Enter password again:
15 Lock out the account after creation? [no]:

```

```

16 Username   : jan
17 Password   : *****
18 Full Name   : Jan Kowalski
19 Uid         : 1002
20 Class      :
21 Groups     : jan
22 Home       : /home/jan
23 Shell      : /bin/sh
24 Locked     : no
25 OK? (yes/no): yes
26 adduser: INFO: Successfully added (jan) to the user database.
27 Add another user? (yes/no): no
28 Goodbye!
29 #

```

Komenda `adduser` działa w sposób interaktywny, prosząc o podanie wartości kolejnych atrybutów użytkownika. Wartości domyślne podane są w nawiasach kwadratowych. Akceptujemy je naciskając klawisz Enter. Jak widać, musimy podać nazwę użytkownika (linia 2). Jego opis jest opcjonalny (linia 3). Numer identyfikacyjny może zostać zaproponowany lub nadany przez komendę (linia 4). Od razu mamy możliwość zdefiniowania grupy podstawowej (linia 5) jak i dodatkowych (linia 6), do których użytkownik będzie należał. W dalszej kolejności definiujemy klasę logowania (linia 7). Wartość kolejnego atrybutu, to podstawowy interpreter poleceń. Lista dostępnych w systemie pochodzi z pliku `/etc/shells`. Możemy również wybrać program `/sbin/nologin`, blokując w ten sposób konto (linia 8). Następnie podajemy ścieżkę dostępu do katalogu domowego użytkownika (linia 9). Następnie decydujemy, że użytkownik będzie autentykował się w systemie z wykorzystaniem hasła (linia 10). Pierwsze hasło użytkownika może być hasłem pustym (linia 11), ale na szczęście odpowiedź *nie* jest domyślna. Możemy wygenerować hasło losowe (linia 12), lub podać własną propozycję (linia 13) oraz w celach weryfikacji jego postaci podać je raz jeszcze (linia 14). Na zakończenie pytanie, czy zablokować konto (linia 15)? Blokada polega na wstawieniu znaku `*` przed postacią zaszyfrowaną hasła w pliku `/etc/master.passwd`. Wartości atrybutów zostają wypisane. Jeśli są poprawne komenda pyta, czy dodajemy kolejnego użytkownika? Jeśli nie, proces definiowania wartości atrybutów zostaje powtórzony z tym, że podane poprzednio wartości są przyjmowane jako domniemane.

W wyniku wykonania powyższej komendy, w plikach konfiguracyjnych pojawiły się następujące wpisy. W pliku `/etc/master.passwd`:

```

1 jan:$1$vFal1Tph$4nKCKatii3xmgT/uTAuvb/:1002:1002::0:0:Jan Kowalski:/home/jan:/bin/sh

```

W pliku `/etc/passwd`:

```

1 jan:*:1002:1002:Jan Kowalski:/home/jan:/bin/sh

```

W pliku `/etc/group`:

```

1 jan:*:1002:

```

Pliku konfiguracyjny `/etc/adduser.conf` tworzymy wywołując komendę `adduser` z opcją `-C`. Przebieg wykonania komendy jest następujący:

```

1 # adduser -C
2 Uid (Leave empty for default):
3 Login group []:
4 Enter additional groups []:
5 Login class [default]:
6 Shell (sh csh tcsh nologin) [sh]: tcsh
7 Home directory [/home/]:
8 Use password-based authentication? [yes]:
9 Use an empty password? (yes/no) [no]:
10 Use a random password? (yes/no) [no]:
11 Lock out the account after creation? [no]:
12 Pass Type : yes
13 Class :
14 Groups :
15 Home : /home/
16 Shell : /bin/tcsh
17 Locked : no
18 OK? (yes/no): y
19 Re-edit the default configuration? (yes/no): n
20 Goodbye!

```

W wyniku jej wykonania, w katalogu */etc* powstaje tekstowy plik konfiguracyjny *adduser.conf* o następującej zawartości:

```

1 # Configuration file for adduser(8).
2 # NOTE: only *some* variables are saved.
3 # Last Modified on Fri Mar 13 09:28:04 UTC 2009.
4
5 defaultLgroup=
6 defaultclass=
7 defaultgroups=
8 passwdtype=yes
9 homeprefix=/home
10 defaultshell=/bin/tcsh
11 udotdir=/usr/share/skel
12 msgfile=/etc/adduser.msg
13 disableflag=
14 uidstart=1003

```

Wartości zmiennych oznaczają:

- *defaultLgroup* – grupę podstawową dodawanego użytkownika.
- *defaultclass* – domyślną klasę logowania.
- *defaultgroups* – grupy dodatkowe użytkownika.
- *passwdtype* – typ hasła. Wartości to: *no* hasło zablokowane (w drugiej kolumnie w linii definiującej użytkownika w pliku */etc/master.passwd* pojawi się \*, *none* hasło puste, *yes* hasło ma zostać podane przez użytkownika, *random* hasło zostanie wygenerowane.



- *homeprefix* – ścieżka dostępu do katalogu, w którym znajdują się katalogi domowe użytkowników.
- *defaultshell* – podstawowy interpreter poleceń.
- *udotdir* – ścieżka dostępu do katalogu, w którym przechowywane są pliki konfiguracyjne interpreterów poleceń (definiujące środowisko pracy) oraz innych aplikacji.
- *msgfile* – ścieżka dostępu do pliku, którego zawartość zostanie wypisana podczas pierwszego podłączania się użytkownika do systemu.
- *disableflag* – blokada konta użytkownika po jego założeniu, jeśli zmienna ma wartość *yes*. Komenda w miejscu zaszyfrowanej postaci hasła umieszcza napis *\*LOCKED\**.
- *uidstart* – numer identyfikacyjny dla następnego dodawanego użytkownika.

Po utworzeniu pliku */etc/adduser.conf* wartości atrybutów dodawanego do systemu użytkownika będą odczytywane z tego pliku. Ale komenda *adduser* posiada opcje, których użycie powoduje, że ustalone przy ich pomocy wartości atrybutów nadpisują te odczytane z pliku konfiguracyjnego. Podstawowe opcje komendy to:

- **-C** – służy do wygenerowania pliku konfiguracyjnego.
- **-d** *ścieżka\_dostępu\_do\_katalogu* – umożliwia zdefiniowanie ścieżki dostępu do katalogu, w którym zostanie utworzony katalog domowy użytkownika.
- **-D** – użycie tej opcji spowoduje, że katalog domowy użytkownika nie zostanie utworzony.
- **-E** – blokuje konto użytkownika poprzez wstawienie w drugim polu linii definiującej użytkownika w pliku */etc/master.passwd* napisu *\*LOCKED\**. Odblokowanie konta następuje w wyniku wykonania komendy *pw unlock*, która jako argument powinna otrzymać nazwę użytkownika lub jego numer identyfikacyjny.
- **-f** *ścieżka\_dostępu\_do\_pliku* – umożliwia zakładanie kont w trybie wsadowym. W pliku podaje się wartości atrybutów użytkowników. Format pliku opisano poniżej.
- **-g** *grupa\_podstawowa* – opcja umożliwia określenie grupy podstawowej użytkownika. Jako wartość opcji można podać nazwę grupy, lub jej numer identyfikacyjny.
- **-G** *grupa [grupa] ...* – opcja umożliwia zdefiniowanie grup, do których należy użytkownik. Jako wartość opcji podawana jest lista nazw lub numerów identyfikacyjnych grup oddzielonych co najmniej jednym białym znakiem.
- **-k** *ścieżka\_dostępu\_do\_katalogu* – umożliwia zdefiniowanie katalogu zawierającego pliki konfiguracyjne interpretera poleceń oraz innych aplikacji. Zostaną one skopiowane do katalogu domowego użytkownika po jego utworzeniu.
- **-L** *klasa\_logowania* – pozwala na określenie klasy logowania użytkownika. Jak wspomniano z klasą logowania związana jest dostępność zasobów systemu.
- **-m** *ścieżka\_dostępu\_do\_pliku* – wartość opcji wskazuje na plik zawierający komunikat powitalny, który zostanie wypisany przy pierwszym podłączeniu użytkownika do systemu.
- **-N** – użycie tej opcji spowoduje, iż komenda *adduser* zignoruje plik konfiguracyjny */etc/adduser.conf*

- **-s** *ścieżka\_dostępu\_do\_pliku* – opcja umożliwia zdefiniowanie innego niż domniemany podstawowego interpretera poleceń.
- **-u** *numer\_użytkownika* – wartość opcji to proponowany numer identyfikacyjny użytkownika w systemie.
- **-w** *typ\_hasła* – opcja umożliwia wybór hasła, które zostanie nadane definiowanemu użytkownikowi. Rozpoznawane są cztery wartości: *no* hasło zablokowane (w drugiej kolumnie w linii definiującej użytkownika w pliku */etc/master.passwd* pojawi się \*), *none* hasło puste, *yes* hasło ma zostać podane przez użytkownika, *random* hasło zostanie wygenerowane jako przypadkowy ciąg znaków.

Użycie komendy *adduser* z opcją **-f** przełącza ją w tzw. wsadowy tryb pracy. Wartością opcji jest ścieżka dostępu do pliku zawierającego wartości atrybutów użytkowników definiowanych w systemie. Plik jest plikiem tekstowym o budowie linikowej. Linie, w których pierwszym nie białym znakiem jest # są ignorowane (traktowane jako komentarz). Jedna linia definiuje wartości atrybutów jednego użytkownika. Separatorem pól jest znak *:*. Ogólna postać linii jest następująca:

```
1 name:uid:gid:class:change:expire:gecos:home_dir:shell:password
```

Pola przechowują:

- *name* – nazwę użytkownika w systemie. Pole to nie może być puste.
- *uid* – propozycję numeru identyfikacyjnego użytkownika w systemie. Jeśli pole to pozostawimy puste, to zostanie on nadany jako pierwszy wolny.
- *gid* – numer grupy podstawowej użytkownika. Jeśli w polu tym nie podano żadnej wartości, to w systemie zostanie utworzona grupa o nazwie identycznej, jak nazwa użytkownika i zostanie ona zdefiniowana jako podstawowa dla tego użytkownika.
- *class* – klasa logowania. Pole może być puste. Wówczas użytkownik zostanie przypisany do klasy zdefiniowanej jako domyślna w pliku */etc/login.defs*.
- *change* – wartość pola określa datę zmiany hasła użytkownika. Zmiana zostanie wymuszona przez system. Akceptowany format to *dd-mm-rr[rr]*, gdzie *dd* oznacza dzień, *mmm* miesiąc jako trzyliterowy skrót nazwy w języku angielskim lub jego numer podany w postaci 2 cyfr, zaś *yy/yy* rok zapisany z użyciem dwóch lub czterech cyfr. Data zmiany hasła może również zostać określona względem bieżącej chwili czasu, a konkretnie momentu definiowania użytkownika w systemie. Wówczas pole to jest zapisane w formacie: *+n[mhdwoy]*, gdzie *n* oznacza liczbę, a *m* – minut, *h* – godzin, *d* – dni, *w* – tygodni, *o* – miesiące, *y* – lat po upływie których hasło będzie musiało zostać zmienione. Przykładowo *+7d* oznacza zmianę hasła po siedmiu dniach od momentu założenia konta, wartość pola w postaci *0-0-0* wymusi zmianę hasła przy pierwszym logowaniu. Pozostawienie pola pustego oznacza, iż nie wprowadzono ograniczenia na maksymalny wiek ważności hasła.
- *expire* – datę ważności konta. Po jej upływie konto zostanie automatycznie zablokowane. Format pola jest identyczny jak pola poprzedniego. Podobnie jak w przypadku określania momentu zmiany hasła, pozostawienie pustego pola oznacza w praktyce, iż ograniczenie nie występuje.

- *gecos* – dodatkowa informacja o użytkowniku. Dopuszcza format w postaci 4 kolumn oddzielonych przecinkiem, a oznaczających kolejno imię i nazwisko użytkownika, numer pokoju, numer telefonu służbowego oraz numer telefonu domowego.
- *home\_dir* – ścieżka dostępu do katalogu domowego. Pozostawienie pola pustego spowoduje, iż ścieżka dostępu do katalogu domowego zostanie zbudowana z połączenia domniemanej ścieżki dostępu do katalogu przechowującego katalogi domowe użytkowników, a katalog użytkownika otrzyma nazwę taką jak definiowany użytkownik. Wpisanie w tym polu wartości */nonexistent* wymusi, aby komenda nie tworzyła katalogu domowego.
- *shell* – ścieżka dostępu do podstawowego interpretera poleceń. Może zostać podana jako ścieżka bezwzględna lub z wykorzystaniem jedynie nazwy interpretera.
- *password* – hasło użytkownika w systemie. Pole to może zawierać postać jawną hasła, które zostanie zaszyfrowane i umieszczone w odpowiednich plikach konfiguracyjnych. Jeśli w wywołaniu komendy *adduser* użyto opcji **-w** z wartością *yes*, a pole pozostawiono puste, komenda zdefiniuje puste hasło. Użycie opcji **-w** z wartością *random* oraz wpisanie hasła w pliku spowoduje, że to hasło zostanie ustawione. Opcja **-w** z wartością *no* lub *none* użyta w wywołaniu komendy spowoduje, że zawartość kolumny zostanie zignorowana.

Test trybu wsadowego komendy *adduser* przeprowadzimy wykorzystując plik o następującej zawartości:

```
1 # name:uid:gid:class:change:expire:gecos:home_dir:shell:password
2 antek:::::Antoni Nowak::sh:
```

Pierwsza linia została umieszczona w pliku jako komentarz i stanowi po prostu ściągę z wartości kolejnych pól. W drugiej linii pojawiły się atrybuty użytkownika. Jego nazwa w systemie to *antek*, dodano jego opis w postaci imienia i nazwiska oraz zdefiniowano podstawowy interpreter poleceń. Pozostałym atrybutom zostaną nadane wartości domyślne. Natępnie wywołujemy komendę *adduser* w trybie wsadowym z opcją **-f** z wartością będącą nazwą pliku zawierającego definicję dodawanego użytkownika oraz z opcją **-w** z wartością *random* wymuszającą ustawienie hasła losowego (użytkownikom, którym nie podano jawnej postaci hasła w ostatniej kolumnie linii). Wynik uruchomienia przedstawiono poniżej. Komenda poinformowała nas, że użytkownik został dodany do systemu oraz podała jego hasło.

```
1 # adduser -w random -f add
2 adduser: INFO: Successfully added (antek) to the user database.
3 adduser: INFO: Password for (antek) is: Jrqvc00V.qSn3Xg
```

## 2.6.2 Zmiana konfiguracji użytkownika

### Aspekty związane z hasłem

W tej rodzinie systemów operacyjnych, dodając użytkownika komendą *adduser* wymuszone zostanie nadanie hasła użytkownikowi. Administrator może od razu zdecydować się na jego zablokowanie, nadanie hasła pustego, zaproponowanie własnego lub zaakceptować wygenerowane przez komendę w postaci ciągu przypadkowych znaków. Wybranie dowolnej, poza pierwszą możliwością powoduje, iż zdefiniowany w systemie użytkownik może się do niego podłączyć. Zmianę hasła użytkownika w systemie może wykonać on sam. Użytkownik *root* może zmienić hasło sobie, lub dowolnemu użytkownikowi w systemie. Istnieje kilka komend służących do zmiany hasła. W

pierwszej kolejności należy wymienić komendę *passwd*. Jest ona najprostsza w użyciu, gdyż wymaga podania postaci jawnej hasła, które następnie szyfruje i zapisuje w odpowiednich plikach konfiguracyjnych. Komendy *chpass* raz *pw* wymagają podania postaci zaszyfrowanej hasła oraz posiadają możliwości zmiany wartości wszystkich atrybutów użytkownika.

Komenda *passwd* w systemie BSD wypada skromnie, jak chodzi o jej możliwości w porównaniu do komendy o tej samej nazwie z dystrybucji Red Hat. Przy jej pomocy jesteśmy w stanie jedynie ustalić nowe hasło. Komenda potrafi za to ustalać hasła lokalne, w systemie Kerberos oraz w NIS<sup>5</sup>. Komenda posiada jedną opcję **-l**. Jej użycie spowoduje, że hasło zostanie zmienione w plikach konfiguracyjnych użytkowników, natomiast nie będzie zmieniane w systemie Kerberos (jeśli jest on uruchomiony w naszym systemie operacyjnym). Zmiana hasła w pliku */etc/master.passwd* spowoduje automatyczne uruchomienie komendy *pwd\_mkdb* w celu uaktualnienia wpisów w bazie danych użytkowników (plik */etc/pwd.db*).

### Wartości pozostałych atrybutów użytkownika

Znacznie większą funkcjonalność posiada komenda *chpass*. Pozwala ona na edycję wartości atrybutów użytkownika zapisanych w bazie danych. Uruchomienie komendy bez argumentu powoduje, że edycji podlegają wartości atrybutów bieżącego użytkownika. Argumentem może być nazwa użytkownika w systemie, którego dane mają zostać zmienione. Użytkownik *root* może edytować atrybuty wszystkich użytkowników w systemie. Pozostali użytkownicy jedynie swoje. Użycie komendy bez opcji powoduje uruchomienie edytora z szablonem atrybutów, których wartości możemy ustalić. Szablon dla użytkownika *root*, który chce zmienić wartości atrybutów zwykłego użytkownika jest następujący:

```

1 #Changing user information for wacek.
2 Login: wacek
3 Password: $1$VvtmRCj.$8UMWBvxfont.s0vqMfpyC81
4 Uid [#]: 1001
5 Gid [# or name]: 1001
6 Change [month day year]:
7 Expire [month day year]:
8 Class:
9 Home directory: /home/wacek
10 Shell: /bin/sh
11 Full Name: Wacław Kowalski
12 Office Location:
13 Office Phone:
14 Home Phone:
15 Other information:

```

Linie, w których pierwszym nie białym znakiem jest *#* oznaczają komentarz. Kolejne linie umożliwiają zmianę nazwy użytkownika w systemie i jego hasła. Postać hasła musi zostać podana w postaci zaszyfrowanej. W dalszej kolejności edycji podlega numer identyfikacyjny użytkownika, grupa podstawowa użytkownika, data następnej zmiany hasła, data wygaśnięcia ważności konta, klasa logowania, ścieżka dostępu do katalogu domowego, podstawowy interpreter poleceń, opis użytkownika z podaniem imienia i nazwiska, lokalizacji pokoju służbowego, oraz telefonów służbowego i domowego. Dodatkowa informacja jest przechowywana w pliku *master.passwd* w polu

<sup>5</sup>Konfiguracje z wykorzystaniem systemów NIS oraz Kerberos zostaną dokładnie omówione w tomie drugim

zawierającym opis użytkownika w kolumnie z imieniem i nazwiskiem. Jej treść nie może zawierać białych znaków, nawet jeśli opis zamknijemy w cudzysłowia lub apostrofach.

Możliwości zwykłego użytkownika są znacznie skromniejsze. Może on zmienić jedynie swój opis w systemie oraz podstawowy interpreter poleceń, co przedstawiono poniżej.

```

1 #Changing user information for wacek.
2 Shell: /bin/sh
3 Full Name: Wacław Kowalski
4 Office Location:
5 Office Phone:
6 Home Phone:
7 Other information:

```

Edycja szablonu odbywa się przy pomocy edytora, który został zdefiniowany jako wartość zmiennej środowiskowej *EDITOR*. W konfiguracji domyślnej jest to edytor *vi*. Zmiany dokonujemy przez przypisanie wartości będącej ścieżką dostępu do edytora, który chcemy wykorzystywać. Sposób zależy od interpretera poleceń, w którym pracujemy. Zagadnienia te zostały dokładnie omówione w podrozdziale 5.3.

Po zakończeniu edycji sprawdzane są wartości wszystkich atrybutów. Niestety tylko pod kątem poprawności składniowej. Przykładowo, wpisanie numeru identyfikacyjnego nieistniejącej grupy, jako podstawowej dla użytkownika nie zostanie na tym etapie wychwycone. Sprawdzana jest natomiast poprawność daty następnej zmiany hasła i ważności konta pod kątem poprawności budowy (np. czy nie jest to 30 dzień lutego, lub nie jest to data sprzed 1 stycznia 1970r.). W przypadku wychwycenia błędu, pojawia się pytanie o ponowną edycję pliku.

Dokonując modyfikacji atrybutów użytkowników zdefiniowanych w systemie lokalnym z konta użytkownika *root*, mamy do dyspozycji kilka opcji, z których najważniejsze to:

- **-a** *lista\_atrybutów* – opcja umożliwia modyfikację wartości wszystkich atrybutów użytkownika w bazie danych. *lista\_atrybutów* musi mieć format linii definiującej użytkownika w pliku *master.passwd*, co przedstawia poniższy przykład:

```

1 # chpass -a 'jan:$1$v0UwGr1K$Sg2vkr8yB2ePW0Hs4Pdnx1:1002:1002::0:0:Jan Kowalski:/home/jan:/bin/
2 chpass: user information updated

```

- **-p** *postać\_zaszyfrowana\_hasła* – umożliwia zdefiniowanie hasła użytkownikowi, którego nazwa pojawiła się jako argument wywołania komendy. Hasło należy podać w postaci zaszyfrowanej.
- **-e** *data\_ważności\_konta* – użycie opcji umożliwia określenie dnia, w którym konto zostanie zablokowane. Datę podajemy w takim formacie, jak w pliku szablonu. Przykładowo:

```

1 # chpass -e "01 01 2015" wacek
2 chpass: user information updated

```

- **-s** *interpreter\_poleceń* – opcja pozwala na zmianę podstawowego interpretera poleceń użytkownika. Jako wartości opcji można użyć bezwzględnej ścieżki dostępu, jak również podać tylko jego nazwę. Przykładowo, wydanie komendy:

```

1 # chpass -s tcsh wacek
2 chpass: user information updated

```

spowoduje, że z pliku */etc/shells* zostanie pobrana bezwzględna ścieżka dostępu w postaci */bin/tcsh* i wpisana do odpowiednich plików konfiguracyjnych. Podanie nazwy interpretera, który nie został zdefiniowany spowoduje błąd wykonania komendy.

Opcja **-s** jest dostępna dla każdego użytkownika systemu.

Komenda *pw* jest określana jako edytor plików konfiguracyjnych użytkowników i grup w systemie. Pracuje w trybie linii komend. Umożliwia administratorowi systemu dodawanie, modyfikowanie oraz usuwanie użytkowników i grupy zdefiniowane lokalnie. Uniwersalność komendy pociąga za sobą skomplikowaną składnię. Pierwsze słowo lub słowa kluczowe służą do ustalenia kontekstu pracy komendy. Buduje się je jako kombinację słów **user** lub **group** ze słowami **add** (dodaj), **del** (usuń), **mod** (zmień charakterystykę), **show** (pokaż atrybuty) lub **next** (pokaż wartość następnego numeru identyfikacyjnego użytkownika lub grupy). Przykładowo: **showuser**, **usershow**, **show user** lub **user show** ustalają ten sam kontekst służący pokazaniu wartości atrybutów użytkownika z pliku */etc/master.passwd*. Argumenty komendy mogą zostać wskazane jako kolejne słowa lub jako wartości odpowiednich opcji komendy (przykładowo komendy: **pw usershow antek** oraz **pw usershow -n antek** dadzą identyczny wynik). Następujące opcje są wspólne dla większości lub wszystkich trybów pracy komendy:

- **-V** *nowy\_katalog\_etc* – pozwala na wyspecyfikowanie ścieżki dostępu do katalogu, w którym znajdują się pliki konfiguracyjne użytkowników i grup (alternatywnego do katalogu */etc*). Użycie tej opcji spowoduje również, że w wyspecyfikowanym katalogu będzie poszukiwany plik konfiguracyjny komendy *pw.conf*, z którego czytane są domyślne wartości atrybutów. Jego domyślnym katalogiem jest katalog */etc* chyba, że użyto opcji **-C**.
- **-C** *plik\_konfiguracyjny* – plikiem konfiguracyjnym komendy *pw* jest plik */etc/passwd*. Użycie tej opcji pozwala na wyspecyfikowanie ścieżki dostępu do alternatywnego pliku konfiguracyjnego, z którego komenda będzie czytała wartości domyślne atrybutów.
- **-q** – wyłącza komunikaty o błędach.
- **-N** – opcja ta jest dostępna ze słowami kluczowymi **add** oraz **modify**. Powoduje, iż komenda raportuje przebieg wykonania operacji ale nie modyfikuje plików konfiguracyjnych użytkowników lub grup.
- **-Y** – tej opcji używa się w konfiguracjach sieciowych. Powoduje ona uruchomienie komendy *make*, która dokonuje modyfikacji zawartości katalogu */etc/yp*.

Najczęściej wykorzystywanymi opcjami dla zmiany wartości atrybutów użytkownika są:

- **-n** *nazwa\_użytkownika* – opcja ta pozwala na wyspecyfikowanie nazwy użytkownika, który ma zostać zdefiniowany w systemie (słowa kluczowe **add** lub **mod**) lub informacja o którym ma zostać wypisana (w przypadku użycia słowa kluczowego **show**).
- **-u** *numer\_identyfikacyjny* – pozwala wyspecyfikować numer identyfikacyjny definiowanego, modyfikowanego lub poszukiwanego użytkownika.
- **-c** *opis\_użytkownika* – umożliwia zdefiniowanie lub modyfikację opisu użytkownika w systemie. Dopuszcza się format w postaci czterech pól oddzielonych przecinkami oznaczającymi kolejno imię i nazwisko, numer pokoju, telefon służbowy oraz telefon domowy.
- **-d** *ścieżka\_dostępu\_do\_katalogu\_domowego* – pozwala na zdefiniowanie lub modyfikację ścieżki dostępu do katalogu domowego użytkownika.

- **-e** *data\_ważności\_konta* – umożliwia zdefiniowanie lub modyfikację daty ważności konta. Datę podajemy zależnie od zdefiniowanego podstawowego języka systemu. W przypadku języka angielskiego ma ona format: *dd-mm-yy/yy*. Miesiąc może zostać podany jako numer dwucyfrowy (01 – styczeń, ...) lub trzyliterowy skrót (trzy pierwsze litery nazwy w języku angielskim: jan – styczeń, ...).
- **-p** *data\_ważności\_hasła* – pozwala na zdefiniowanie lub modyfikację daty ważności hasła. Upłynięcie daty ważności spowoduje, iż system wymusi na użytkowniku zmianę hasła przy pierwszej próbie autentykacji z jego wykorzystaniem. Może to być podłączanie się do systemu lub autentykacja przy okazji użycia przykładowo komendy *chsh* lub *chfn*. Format daty jest identyczny jak w przypadku użycia opcji **-e**. Podanie daty ważności hasła w postaci *0-0-0* spowoduje wymuszenie zmiany hasła przy najbliższej autentykacji, w szczególności podłączaniu się do systemu.
- **-g** *grupa\_podstawowa* – użycie opcji pozwala na zdefiniowanie lub zmianę grupy podstawowej użytkownika. Jako wartość opcji można podać numer identyfikacyjny lub nazwę grupy.
- **-G** *lista\_grup\_dodatkowych* – wartością opcji jest lista nazw lub numerów grup oddzielonych przecinkiem. Użytkownik, którego nazwa została podana jako argument wywołania komendy *pw* lub wartość opcji **-n** stanie się ich członkiem.
- **-L** *klasa\_logowania* – pozwala na zdefiniowanie lub zmianę klasy logowania użytkownika.
- **-m** – opcja ta wymusza utworzenie katalogu domowego użytkownika. Jest domyślnie wykorzystywana podczas dodawania użytkownika do systemu. Może zostać również użyta w celu przeniesienia katalogu domowego użytkownika. Istotna jest kwestia plików konfiguracyjnych. Podczas definiowania nowego użytkownika czyli użyciu słowa kluczowego **useradd** są one kopiowane z domyślnego katalogu */usr/share/skel* do utworzonego katalogu domowego. W przypadku przenoszenia katalogu domowego, co odpowiada użyciu słowa kluczowego **usermod**, pliki konfiguracyjne będą kopiowane, a więc istniejące nie zostaną nadpisane.
- **-M** *prawa\_dostępu* – opcja ta jest wykorzystywana z opcją **-m** i umożliwia wyspecyfikowanie praw dostępu do tworzonego katalogu domowego. Jeśli opcja zostanie pominięta, to prawa dostępu zostaną nadane w opciu o wartość *umask* ze środowiska procesu rodzica komendy *pw* (najczęściej jest to interpreter poleceń użytkownika *root*). Wartość podajemy w zapisie ósemkowym.
- **-k** *katalog\_plików\_konfiguracyjnych* – opcja umożliwia wyspecyfikowanie ścieżki dostępu do katalogu, w którym znajdują się pliki konfiguracyjne interpreterów poleceń oraz innych aplikacji. Zostaną one skopiowane do katalogu domowego definiowanego użytkownika.
- **-s** *interpreter\_poleceń* – wartością opcji jest ścieżka dostępu lub nazwa interpretera poleceń, który zostanie podstawowym interpreterem poleceń użytkownika.
- **-h** *deskryptor\_pliku* – opcja ta umożliwia stworzenie interfejsu umożliwiającego nadanie hasła użytkownikowi z wykorzystaniem mechanizmu przekierowania standardowego wejścia, co znajduje praktyczne zastosowanie w skryptach np. dla interpreterów poleceń. Jako wartość opcji podawany jest deskryptor pliku z którego hasło zostanie odczytane. Wywołanie komendy *pw* z opcją **-h 0**, powoduje, że hasło czytane będzie ze standardowego wejścia, jednokrotnie, bez jego weryfikacji:

```

1 # pw usermod -h 0 -n jan
2 new password for user jan:

```

Użycie jako wartości opcji znaku `-` spowoduje, że hasło zostanie zablokowane przez wstawienie znaku `*` w drugim polu linii opisującej użytkownika w pliku `/etc/master.passwd` oraz w bazie danych użytkowników.

- **-H deskryptor\_pliku** – umożliwia przeczytanie postaci zaszyfrowanej hasła z pliku o deskryptorze będącym wartością opcji. Działa zatem podobnie do opcji **-h**, jednak z tą różnicą, że hasło musi zostać podane w postaci zaszyfrowanej, w formacie *\$identyfikator\$śól\$hasło*. Komenda może zostać użyta w następującej postaci:

```

1 # pw usermod jan -h 0 < haslo.txt

```

*haslo.txt* jest nazwą pliku zawierającego postać zaszyfrowaną hasła.

Korzystając z komendy *pw* ze słowem kluczowym **useradd** możemy utworzyć w systemie użytkownika o powtarzającym się numerze identyfikacyjnym. Jeśli jest to działanie celowe, należy użyć opcji **-o**, gdyż w przeciwnym przypadku komenda zwróci błąd. Przypisywanie różnym użytkownikom zdefiniowanym w systemie tego samego numeru identyfikacyjnego stosuje się np. jeśli chcemy, aby ta sama osoba miała możliwość podłączania się do systemu w różnych kontekstach. Chodzi tu o przynależność do różnych grup, posiadanie różnych katalogów domowych, środowiska pracy czy interpretera poleceń. Jednak lista plików w systemie będących jej własnością jest taka sama.

Komenda *pw* posiada plik konfiguracyjny, w którym przechowywane są wartości domyślne atrybutów. Jest to plik tekstowy, o nazwie *pw.conf* i znajduje się w katalogu `/etc`. Tworzymy go wywołując komendę *pw* ze słowem kluczowym **useradd**, korzystając z opcji **-D**. Nie możemy wówczas korzystać z opcji **-n** oraz **-u**, zaś użycie opcji **-D** modyfikuje znaczenie następujących opcji:

- **-D** – użycie opcji powoduje, że wartości domyślne atrybutów zostają zapisane do pliku `/etc/pw.conf`. Użycie opcji **-C** pozwoli wskazać inną lokalizację pliku konfiguracyjnego.
- **-b ścieżka\_dostępu\_do\_katalogu** – opcja pozwala wskazać katalog, w którym będą tworzone katalogi domowe użytkowników. Zazwyczaj jest to katalog `/home`.
- **-e liczba\_dni** – pozwala określić liczbę dni, które upłyną od utworzenia konta użytkownika w systemie do jego zablokowania. Wartość opcji jest liczbą. W poznanych dotychczas komendach systemu BSD podawano datę utraty ważności konta użytkownika.
- **-p liczba\_dni** – służy do podania liczby dni, które upłyną od momentu ustawienia hasła do jego przedawnienia. Określony zatem zostaje maksymalny wiek hasła. Również w tym przypadku podajemy liczbę dni, a nie datę.
- **-g grupa\_podstawowa** – umożliwia określenie grupy podstawowej dla definiowanego w systemie użytkownika. Jeśli chcemy, aby każdy definiowany użytkownik tworzył grupę, która będzie jego grupą podstawową, w wywołaniu komendy *pw* opcję **-g** należy użyć z pustą wartością (`-g`).
- **-G lista\_grup** – pozwala zdefiniować grupy, których członkiem zostanie definiowany użytkownik. Lista zawiera nazwy lub numery identyfikacyjne grup oddzielone przecinkiem.



- **-L** *klasa\_logowania* – definiuje klasę logowania użytkownika.
- **-k** *ścieżka\_dostępu\_do\_katalogu* – pozwala zdefiniować ścieżkę dostępu do katalogu zawierającego pliki konfiguracyjne interpretera poleceń oraz innych aplikacji, które zostaną skopiowane do katalogu domowego definiowanego użytkownika.
- **-u** *min,max* – pozwala zdefiniować zakres, z którego będą przydzielane numery identyfikacyjne definiowanych użytkowników. W dystrybucji FreeBSD teoretycznie jest to przedział obustronnie domknięty od 1001 do 32000. Wartości mniejsze od 100 i większe od 32000 są zarezerwowane dla użytkowników pełniących w systemie specjalne funkcje lub dla usług.
- **-i** *min,max* – pozwala zdefiniować zakres, z którego będą przydzielane numery identyfikacyjne definiowanych w systemie grup. Podobnie jak w przypadku numerów identyfikacyjnych użytkowników, za dostępne do wykorzystania przyjmuje się wartości z przedziału od 1001 do 32000 włącznie.
- **-w** *metoda* – rozwiązuje kwestię sposobu ustalania hasła podczas definiowania użytkownika w systemie. Opcja posiada cztery możliwe wartości. *no* spowoduje, że dla każdego definiowanego w systemie użytkownika jego hasło zostanie zablokowane w wyniku umieszczenia znaku \* w drugim polu linii opisującej użytkownika w pliku */etc/passwd* oraz w bazie danych użytkowników. *yes* wymusi, aby hasło było identyczne z nazwą użytkownika w systemie. *none* będzie ustawiało hasło puste, zaś *random* generowało hasło w postaci ciągu przypadkowych znaków.

Budowę pliku */etc/pw.conf* przedstawiono poniżej. Usunięto z niego puste linie, a w celu poprawienia czytelności linie zawierające instrukcje odsunięto od lewego marginesu. Podobnie, jak w pozostałych plikach konfiguracyjnych przyjmuje się, że komentarzem jest fragment linii rozpoczynający się znakiem #, a kończący przejściem do nowej linii.

```

1  #
2  # pw.conf - user/group configuration defaults
3  #
4  # Password for new users? no=nologin yes=loginid none=blank random=random
5  defaultpasswd = "no"
6  # Reuse gaps in uid sequence? (yes or no)
7  reuseuids = "no"
8  # Reuse gaps in gid sequence? (yes or no)
9  reusegids = "no"
10 # Path to the NIS passwd file (blank or 'no' for none)
11 nispasswd =
12 # Obtain default dotfiles from this directory
13 skeleton = "/usr/share/skel"
14 # Mail this file to new user (/etc/newuser.msg or no)
15 newmail = "no"
16 # Log add/change/remove information in this file
17 logfile = "/var/log/userlog"
18 # Root directory in which $HOME directory is created
19 home = "/home"
20 # Mode for the new $HOME directory, will be modified by umask
21 homemode = 0777
22 # Colon separated list of directories containing valid shells

```

```

23     shellpath = "/bin"
24 # Comma separated list of available shells (without paths)
25     shells = "sh","csh","tcsh"
26 # Default shell (without path)
27     defaultshell = "sh"
28 # Default group (leave blank for new group per user)
29     defaultgroup = ""
30 # Extra groups for new users
31     extragroups =
32 # Default login class for new users
33     defaultclass = ""
34 # Range of valid default user ids
35     minuid = 1000
36     maxuid = 32000
37 # Range of valid default group ids
38     mingid = 1000
39     maxgid = 32000
40 # Days after which account expires (0=disabled)
41     expire_days = 0
42 # Days after which password expires (0=disabled)
43     password_days = 0

```

Na zakończenie warto jeszcze wspomnieć o zmianach opisu użytkownika oraz jego podstawowego interpretera poleceń. Zmian tych może dokonać zarówno użytkownik *root* w odniesieniu do siebie oraz innych użytkowników w systemie jak również zwykły użytkownik oczywiście tylko w stosunku do siebie. Zmiany te najprościej przeprowadzić komendami *chfn* oraz *chsh*. Komendy te w domyślnej konfiguracji nie wymagają autentykacji użytkownika. W dystrybucji FreeBSD obie komendy uruchamiają edytor, do którego wczytywany jest identyczny szablon zawierający opis oraz aktualne wartości atrybutów. Domyślnym edytorem jest edytor *vi*. Można go zmienić, ustalając ścieżkę dostępu do preferowanego edytora jako wartość zmiennej środowiskowej *EDITOR*. Szablon ma następujący format:

```

1 #Changing user information for jan.
2 Shell: /bin/sh
3 Full Name: Jan Kowalski
4 Office Location:
5 Office Phone:
6 Home Phone:
7 Other information:

```

Jak widać, jest on identyczny z szablonem wykorzystywanym przez komendę *chpass*. Skąd zatem to podobieństwo. Otóż jest to ta sama komenda, a jej kod jest zapisany w pliku, który występuje w drzewie katalogów pod kilkoma różnymi nazwami. Jest to efekt wykorzystania tzw. dowiązania twardego, którego mechanizm omówiono w rozdziale 3.2.1.

### 2.6.3 Monitorowanie podłączania się do systemu

W dystrybucji FreeBSD istnieje kilka programów narzędziowych pozwalających śledzić zarówno bieżące sesje użytkowników, jak również historię połączeń do systemu. Listowanie użytkowników aktualnie podłączonych do systemu możemy przeprowadzić za pomocą komend *who* oraz *w*.

Historię połączeń do systemu udostępnia komenda *last* oraz komenda *who*. Komendy te może wykonywać każdy użytkownik systemu. Administrator systemu ma dodatkowo wgląd do plików dziennika, które znajdują się w katalogu */var/log*. Jak widać, zestaw narzędzi do monitorowania aktywności użytkowników w systemie jest identyczny jak w dystrybucji Fedora. Oczywiście pojawiają się pewne różnice w detalach, np. formacie dostarczanej informacji, czy opcjach wywołania.

### Komenda *who*

W najprostszym przypadku komendę *who* można uruchomić bez opcji i bez argumentu. Wówczas informacja dostarczana jest z pliku */var/log/utmp*. Poniżej przedstawiono przykład działania komendy:

1	# who -HT			
2	NAME	S LINE	TIME	FROM
3	root	- ttyv0	Mar 20 15:05	
4	antek	+ ttyv1	Mar 20 15:05	
5	wacek	+ ttyp0	Mar 20 13:04	(141.152.99.13)
6	wacek	+ ttyp1	Mar 20 13:46	(141.152.99.13)

Komendę wywołano z dwoma opcjami. Opcja **-H** powoduje wypisanie linii nagłówkowej. Dzięki niej znamy zawartość kolejnych kolumn. Oznaczają one kolejno:

1. Kolumna pierwsza zawiera nazwę użytkownika w systemie, którego sesję opisuje dana linia.
2. Pojawienie się kolumny S niosącej informację, czy użytkownik w danej sesji pozwala na wypisywanie komunikatów na ekranie przesyłanych przez innych użytkowników, zawdzięczamy opcji **-T**.
3. W kolumnie trzeciej znajduje się opis terminala. Opis zaczyna się od skrótu *tty*. Następnie pojawia się jedna litera, która informuje, czy jest to połączenie lokalne, oznaczane literą *v*, czy zdalne, które oznacza się literą *p*. Ostatnia część to numer terminala. Numeracja jest oddzielna dla terminali połączeń lokalnych i zdalnych.
4. Kolumna czwarta zawiera moment połączenia się do systemu w formacie *miesiąc-dzień godzina:minuta* jeśli rok połączenia do systemu jest bieżącym. Jeśli natomiast połączenie miało miejsce wcześniej, data zostaje rozszerzona o rok połączenia.
5. Zawartość ostatniej kolumny zależy od tego, czy użytkownik połączył się do systemu lokalnie, czy zdalnie. W pierwszym przypadku zawartość jest pusta. W drugim pojawia się w niej adres IP lub symboliczny hosta, z którego nastąpiło połączenie.

Pośród opcji komendy *who* należy wymienić:

- **-H** – powoduje wypisanie nagłówka informującego o zawartości poszczególnych kolumn.
- **-m** – użycie tej opcji ogranicza wypisywaną informację jedynie do sesji, których terminal jest skojarzony ze standardowym wejściem (sesja z bieżącego terminala).
- **-q** – format szybki, informujący o nazwach połączonych użytkowników oraz ich liczbie, w postaci:

```

1 # who -q
2 jan          wacek
3 # users = 2

```

- **-s** – opcja domyślna. Informacja jest wypisywana bez nagłówka i obejmuje nazwę użytkownika, nazwę terminala, czas podłączenia się do systemu oraz w przypadku podłączenia zdalnego adres IP lub symboliczny hosta, z którego nastąpiło podłączenie.
- **-T** – w wyniku użycia tej opcji pojawia się informacja o tym, czy w danej sesji użytkownik dopuszcza wypisywanie komunikatów przesyłanych przez innych użytkowników (np. komendą *write*) na terminalu. Znak + oznacza dopuszczenie, – zabronienie, zaś ? błąd przy określaniu stanu terminala. Użytkownik może zmieniać stan terminala przy pomocy komendy *mesg*. Użycie jej ze słowem kluczowym **yes** spowoduje zezwolenie, zaś ze słowem **no** zabronienie wypisywania komunikatów. Wiąże się to ze zmianą praw dostępu do pliku reprezentującego terminal w katalogu */dev*.
- **-u** – użycie opcji spowoduje wypisanie informacji o czasie bezczynności użytkowników podłączonych do systemu, rozumianego jako czasu, który upłynął od wydania ostatniej komendy. Format informacji to *godzina:minuta*. Po rzekroczeniu 24 godzin w kolumnie *IDLE* pojawi się słowo *old*.

```

1 # who -uH
2 NAME          LINE      TIME          IDLE  FROM
3 jan           ttyv0    Mar 27 12:27  02:04 (141.152.99.13)
4 wacek        ttyv1    Mar 27 12:27    .    (141.152.99.13)

```

Argumentem wywołania komendy *who* może być nazwa pliku, z którego odczytana zostanie informacja o sesjach użytkowników w systemie. Jeśli jako argument zostanie podany plik *wtmp* z katalogu */var/log* to uzyskamy w ten sposób informację o historii podłączeń użytkowników do systemu, restartów systemu i jego awariach. Fragment informacji uzyskanej w ten sposób przedstawiono poniżej:

```

1 # who -HT /var/log/wtmp
2 NAME          S LINE      TIME          FROM
3 root          - ttyv0    Feb 21 09:34
4 root          - ttyv0    Feb 21 09:53
5 root          - ttyv0    Feb 21 09:57
6 wacek        + ttyv0    Feb 21 10:20 (141.152.99.19)
7 wacek        - ttyv0    Feb 19 11:22
8 root          - ttyv0    Feb 17 10:22
9 .....

```

Rejetrowanie zdarzeń związanych z podłączaniem się i odłączaniem użytkowników, restartem systemu powoduje gwałtowny wzrost rozmiaru pliku */var/log/wtmp*. Stąd prowadzi się politykę cyklicznego kopiowania zawartości do plików archiwalnych, których nazwa składa się z nazwy pliku oryginalnego i dołączonego do nazwy numeru archiwum. Numer jest z zakresu od 1 do 6. Im wyższy, tym starsze zdarzenia są przechowywane w pliku.

**Komenda *w***

Komenda *w* dostarcza informacji o użytkownikach aktualnie podłączonych do systemu oraz dodatkowo pozwala poznać ogólny stan systemu. Użycie jej bez opcji i argumentów daje następujący wynik:

```

1 # w
2 3:06PM up 2 days, 6 hrs, 4 users, load averages: 0.00, 0.00, 0.00
3 USER      TTY      FROM          LOGIN@  IDLE WHAT
4 root      v0        -             3:05PM   1 -csh (csh)
5 antek     v1        -             3:05PM   - -sh (sh)
6 wacek     p0        141.152.99.13 1:04PM   - w
7 wacek     p1        141.152.99.13 1:46PM   25 more

```

Dostarczona informacja składa się z dwóch fragmentów. Pierwsza linia to właśnie ogólna informacja o systemie. Można ją uzyskać komendą *uptime*. Tworzą ją właściwie cztery pola, które oznaczają:

1. Bieżący czas systemowy.
2. Po słowie *up* pojawia się informacja ile czasu minęło od inicjalizacji systemu. W naszym przypadku są to 2 dni i 6 godzin. Jeśli czas ten jest krótszy niż 24 godziny, to pojawi się on w formacie *godziny:minuty*.
3. W polu trzecim znajduje się informacja o liczbie otwartych sesji (podłączeń, a nie liczbie podłączonych użytkowników).
4. Ostatnie pole informuje o średnim obciążeniu systemu. Miarą jest średnia liczba procesów stojących w kolejce gotowych do wykonania wyznaczona z ostatniej minuty, ostatnich 5 i 15 minut.

Druga część informacji dostarczanej przez komendę *w* w formie domyślnej składa się z sześciu kolumn, które oznaczają:

Opcje dostępne w komendzie *w* umożliwiają dostarczenie bardziej szczegółowych informacji. Do podstawowych należą:

- **-d** – powoduje wypisanie informacji o wszystkich procesach skojarzonych z terminalem, a nie tylko o ostatnim w hierarchii. Poniżej zamieszczono przykład. Użytkownik *jan*, jest podłączony z konsoli *p0* (zdalnej), procesy związane z terminalem mają numery identyfikacyjne: 34395 - interpreter poleceń, 34423 - użytkownik czyta manual komendy *w*, 34424 oraz 34426 to procesy uruchomione przez komendę *man* w celu rozpakowania oraz wypisania dokumentacji w postaci sformatowanej.

```

1 # w -d
2 3:23PM up 10 days, 6:17, 3 users, load averages: 0.00, 0.00, 0.00
3 USER      TTY      FROM          LOGIN@  IDLE WHAT
4           8165      login [pam] (login)
5           34429      -csh (csh)
6 root      v0        -             3:23PM   - -csh (csh)
7           34395      -sh (sh)
8           34423      man w

```

9		34424	sh -c /usr/bin/zcat /usr/share/man/cat1/w.1.gz	more
10		34426	more	
11	jan	p0	141.152.99.13	3:21PM 1 more
12		34403	-sh (sh)	
13		34405	su -	
14		34406	-su (csh)	
15		34433	w -d	
16	wacek	p1	141.152.99.13	3:21PM - w -d

- **-h** – użycie tej opcji spowoduje, że nagłówek informujący o zawartości kolumn nie zostanie wypisany.
- **-i** – informacja o podłączonych do systemu użytkownikach zostanie wypisana w/g rosnącego czasu bezczynności.
- **-n** – w przypadku połączeń zdalnych blokuje rozwiązywanie adresu. Domyślnie komenda *w* stara się znaleźć adres symboliczny hosta, z którego nastąpiło połączenie. Jeśli nie jest to możliwe lub jeśli użyto tej opcji, wypisany zostaje adres IP.

Argumentem komendy *w* może być nazwa użytkownika. Użycie go powoduje, że wypisywana jest jedynie informacja dotyczące tego użytkownika.

### Komenda *last*

Omówione powyżej komendy *who* oraz *w* wykorzystywane są najczęściej do uzyskiwania informacji o użytkownikach aktualnie podłączonych do systemu. Komenda *w* użyta z argumentem będącym ścieżką dostępu do pliku */var/log/wtmp*, wypisuje informacje o historii połączeń. Plik ten jest domyślnym dla komendy *last*. Komenda przegląda go od końca, wypisując zawarte w nim informacje w odwrotnym porządku chronologicznym. Uruchomiona bez opcji dostarcza całą zawartość pliku. Opcja **-n** jest wykorzystywana do określenia liczby ostatnich zdarzeń, które mają zostać wypisane. Przykładowo:

```

1 # last -n 10
2 wacek      ttyt1    141.152.99.13    Fri Mar 20 13:46    still logged in
3 wacek      ttyt0    141.152.99.13    Fri Mar 20 13:04    still logged in
4 wacek      ttyt1    141.152.99.13    Fri Mar 20 12:16 - 12:26    (00:10)
5 wacek      ttyt0    141.152.99.13    Fri Mar 20 12:01 - 12:27    (00:25)
6 wacek      ttyt0    141.152.99.13    Thu Mar 19 11:01 - 17:01    (06:00)
7 reboot     ~
8 shutdown   ~
9 wacek      ttyt0    141.152.99.13    Mon Mar 16 13:11 - 15:37    (02:26)
10 root       ttyv0
11 reboot     ~
12
13 wtmp begins Sat Feb 21 09:34:34 UTC 2009

```

Jak widać format dostarczanej przez komendę *w* informacji jest w dystrybucji FreeBSD jest praktycznie identyczny, jak w systemach z rodziny Red Hat. Dla przypomnienia, w pierwszej kolumnie pojawia się nazwa użytkownika, który w systemie pracował lub pracuje. W pliku zapisywane są również rekordy opisujące inne zdarzenia, jak np. zamknięcie systemu (*shutdown*), czy

jego inicjalizacja (*reboot*). Jeśli opis dotyczy sesji użytkownika, to w kolumnie drugiej pojawia się nazwa terminala, z którego podłączył się on do systemu. Jeśli w nazwie, po przedrostku *tty* występuje literka *p*, to mamy do czynienia z połączeniem zdalnym (*pseudoterminal*), jeśli literka *v*, to połączenie nastąpiło z konsoli. W przypadku innych zdarzeń kolumna druga nie zawiera żadnych informacji. Kolumna trzecia zawiera uzupełnienie opisu połączeń zdalnych o adres IP lub symboliczny hosta, z którego nastąpiło połączenie. Kolumna czwarta zawiera dokładną datę i czas połączenia. Kolejna kolumna zawiera informacje o czasie zakończenia sesji. Dla rekordów opisujących zdarzenia inne niż sesja użytkownika, nie ma ona żadnej treści. Podobnie ostatnia kolumna. Dla sesji użytkowników zawiera informację o czasie jej trwania. Domyślny format to *godziny:minuty*.

W ostatniej linii zostaje wypisana informacja o dacie utworzenia pliku */var/log/wtmp*, czyli w praktyce dacie najstarszego zachowanego zdarzenia. W dystrybucji FreeBSD działania na plikach dziennika wykonuje się z wykorzystaniem komendy *newsyslog*. Jej wywołanie zostało umieszczone w tablicy *crontab* demona zegarowego użytkownika *root*. Jego konfiguracja została omówiona w podrozdziale 4.3.

Argumentem komendy *last* może być nazwa użytkownika w systemie. Uruchomiona w ten sposób komenda wypisze jedynie dostępne informacje o sesjach tego użytkownika. Do najczęściej wykorzystywanych opcji komendy *last* należy zaliczyć:

- **-d data** – użycie tej opcji umożliwia wypisanie informacji o wszystkich użytkownikach, którzy podłączyli się do systemu w momencie określonym wartością *data* opcji. Format daty jest następujący: *[[CC]YY][MMDD]hhmm[.SS]*, gdzie
  - *CC* – oznacza pierwsze dwie cyfry roku (stulecie).
  - *YY* – ostatnie dwie cyfry roku. Jeśli pierwsze *CC* nie zostały wyspecyfikowane, to ich wartość określa się na podstawie drugich *YY*. Liczba z przedziału od 69 do 99 implikuje wartość *CC* 19, zaś z przedziału od 00 do 68 wartość 20.
  - *MM* – miesiąc, wartość z przedziału od 1 do 12.
  - *DD* – dzień, wartość z przedziału od 1 do 31.
  - *hh* – godzina, wartość z przedziału od 0 do 23.
  - *mm* – minuta, wartość z przedziału od 0 do 59.
  - *SS* – sekunda, wartość z przedziału od 0 do 59.
- **-f ścieżka\_dostępu\_do\_pliku** – umożliwia podanie ścieżki dostępu do pliku, na którym będzie pracować komenda, zamiast na dominiemanym */var/log/wtmp*.
- **-h host** – wartość opcji to adres IP lub symboliczny hosta. Opcja umożliwia filtrowanie połączeń do systemu, które miały miejsce z hosta o podanym adresie.
- **-n liczba\_zdarzeń** – opcja umożliwia wypisanie informacji o takiej liczbie najmłodszych zdarzeń, jak podana wartość opcji.
- **-s** – powoduje, że czas trwania sesji jest raportowany w sekundach, a nie w formacie *godziny:minuty:sekundy*.
- **-t tty** – opcja umożliwia filtrowanie sesji użytkowników przeprowadzonych z określonego terminala. Nazwę terminala możemy podać jako pełną lub w sposób krótki, bez przedrostka *tty*. Stąd wywołania: *last -t ttyv0* oraz *last -t v0* dają identyczny efekt.
- **-w** – użycie opcji sprawia, że czas trwania sesji jest podawany w formacie *godziny:minuty:sekundy*.
- **-y** – zmiana format kolumny opisującej datę rozpoczęcia sesji przez podanie roku.

**Katalog `/var/log`**

W katalogu tym system przechowuje pliki dziennika. Jego zawartość dla domyślnej konfiguracji systemu FreeBSD przedstawiono poniżej:

```

1 # ls /var/log
2 auth.log      ipfw.today    maillog.4.bz2  security      sendmail.st.7
3 cron          ipfw.yesterday maillog.5.bz2  sendmail.st   setuid.today
4 cron.0.bz2    lastlog       maillog.6.bz2  sendmail.st.0 slip.log
5 cron.1.bz2    lpd-errs      maillog.7.bz2  sendmail.st.1 userlog
6 cron.2.bz2    maillog        messages       sendmail.st.2 wtmp
7 cron.3.bz2    maillog.0.bz2 messages.0.bz2  sendmail.st.3 wtmp.0
8 debug.log     maillog.1.bz2 mount.today    sendmail.st.4 xferlog
9 dmesg.today   maillog.2.bz2 pf.today       sendmail.st.5
10 dmesg.yesterday maillog.3.bz2 ppp.log        sendmail.st.6

```

Obserwując nazwy plików możemy stwierdzić, że istotne fragmenty funkcjonalne systemu w tym usługi mają własne dzienniki. Pełną listę tzw. kanałów, czyli etykiet komunikatów zapisywanych do jednego pliku dziennika, dostępnych w dystrybucji FreeBSD wraz z ich krótką charakterystyką zamieszczono w tabeli 4.4.2. Dla naszych potrzeb ograniczymy się jedynie do pliku *auth.log* zawierającego publicznie dostępne informacje o podłączeniach do systemu i zmianach tożsamości. Jego fragment przedstawiono poniżej:

```

1 Feb 21 09:34:32 newsyslog[591]: logfile first created
2 Feb 21 09:34:51 login: login on ttyv0 as root
3 Feb 21 10:10:05 sshd[729]: error: PAM: authentication error for root from 141.152.99.13
4 Feb 21 10:13:55 sshd[745]: error: PAM: authentication error for root from 141.152.99.13
5 Feb 21 10:20:21 sshd[793]: Accepted keyboard-interactive/pam for wacek from 141.152.99.13 po
6 Feb 21 10:21:12 sshd[802]: error: PAM: authentication error for root from 141.152.99.13
7 Feb 21 10:21:19 sshd[802]: error: PAM: authentication error for root from 141.152.99.13
8 Feb 21 10:22:17 login: login on ttyv0 as wacek
9 Feb 21 10:22:29 su: BAD SU wacek to root on /dev/ttyv0
10 Feb 21 10:22:52 login: login on ttyv0 as root
11 Feb 21 10:22:52 login: ROOT LOGIN (root) ON ttyv0
12 Feb 21 10:22:57 shutdown: halt by root:

```

Jest to plik tekstowy, w którym jedna linia opisuje jedno zdarzenie, w tym przypadku z kanału *auth*. Pierwsza część linii zawiera datę zdarzenia, a w drugiej pojawia się jego opis. W linii 1 znajduje się opis zdarzenia o dacie utworzenia pliku. linia 2 informuje o poprawnym podłączeniu się do systemu przez użytkownika *root* z konsoli (terminal *ttyv0*). Linie 3 oraz 4 zawierają informację o nieudanym podłączeniu się do systemu użytkownika *root* z hosta o podanym adresie IP. Przypomnijmy, że domniemana konfiguracja nie pozwala na bezpośrednie, zdalne podłączanie się do systemu użytkownikowi *root*. Podłączamy się jako dowolny użytkownik ale należący do grupy *wheel*, a następnie przez zmianę tożsamości komendą *su*, przełączamy się na użytkownika *root*. Linia 5 informuje o zajściu zdarzenia polegającego na zdalnym podłączeniu się do systemu użytkownika *wacek* z hosta o podanym adresie. Kolejne dwie linie, to nieudana próba podłączenia zdalnego podłączenia się do systemu użytkownika *root*. Linia 8 opisuje podłączenie się do systemu użytkownika *wacek*. To z jego konta dokonano pierwszej zmiany tożsamości na użytkownika *root*. Niestety pierwsza była nieudana, gdyż podano niepoprawne hasło (linia 9). Za drugim razem



podano poprawne hasło i użytkownik *root* do systemu się podłączył (linie 10 i 11). Następnie użytkownik *root* dokonał zamknięcia systemu, o czym informuje zdarzenie z linii 12.

Ze względu na dużą liczbę zdarzeń zachodzących w eksploatowanym systemie konieczne jest zapewnienie cyklicznej archiwizacji plików dziennika. W dystrybucji FreeBSD mamy do dyspozycji program narzędziowy *newsyslog* opisany dokładnie w podrozdziale 4.4.2. Posiada on plik konfiguracyjny, w którym definiujemy częstotliwość archiwizacji oraz liczbę przechowywanych plików archiwalnych. Program automatycznie usuwa najstarszy plik archiwum, przesuwając na jego miejsce poprzedni. Aktualny plik dziennika staje się pierwszym plikiem archiwalnym. Samo archiwum w tym momencie staje się plikiem z jednym wpisem informującym o momencie jego utworzenia.

Domyślne konfiguracje dzienników w dystrybucjach Red Hat oraz FreeBSD różnią się znacznie. Również w systemie FreeBSD mamy szerokie możliwości ingerencji w rodzaj oraz istotność dziennikowanej informacji. Zagadnieniu temu poświęcono podrozdział 4.4.2.

### 2.6.4 Ograniczanie dostępu do systemu

Poniżej przedstawione zostaną podstawowe mechanizmy pozwalające ograniczyć dostęp do systemu pojedynczemu użytkownikowi. W dalszej kolejności zajmiemy się bardziej precyzyjnymi mechanizmami konfigurowanymi w pliku */etc/login.access*. Zaawansowane mechanizmy pozwalające na kontrolę sposobu autentykacji użytkowników w systemie są przedmiotem podrozdziału 2.9 poświęconemu mechanizmom PAM, rozdziału 2.10 oraz rozdziału 7 opisującemu podstawowe zagadnienia konfiguracji sieciowych.

#### Mechanizmy proste

Najprostszy sposób uniemożliwienia podłączania użytkownika do systemu jest unieważnienie hasła. Można to zrobić edytując plik */etc/master.passwd* przy pomocy komendy *vipw*. Hasło unieważnia się wstawiając znak *\** przed postacią zaszyfrowaną hasła, w drugim polu linii opisującej wybranego użytkownika. Natomiast komenda:

```
1 # pw usermod -w no -n wacek
```

usunie postać zaszyfrowaną hasła, a w jej miejsce wstawi znak *\**. Stąd aby przywrócić użytkownikowi możliwość podłączania się do systemu, musimy ponownie ustawić mu hasło.

Innym sposobem jest skorzystanie z odpowiedniego interpretera poleceń. Dostępny w systemie interpreter *nologin*, znajdujący się w katalogu */sbin* nie umożliwi użytkownikowi podłączenia się do systemu, a jedynie wypisze stosowny komunikat na ekranie. Można również napisać własny program, który wykona stosowne czynności i zakończy swoje działanie, co spowoduje powrót do programu logującego, który będzie monitorował o podanie nazwy użytkownika. Należy pamiętać, aby ze względów bezpieczeństwa systemu ścieżkę dostępu do tego programu dopisać do pliku */etc/shells*.

Przedstawione powyżej sposoby ograniczania dostępu do systemu są efektywne, jeśli mają dotyczyć maksymalnie kilku użytkowników. Często zachodzi jednak konieczność zablokowania dostępu do systemu jego wszystkim użytkownikom. W takiej sytuacji blokowanie hasła lub zmiana podstawowego interpretera poleceń każdemu użytkownikowi z osobna jest praktycznie niewykonalne. Stąd stworzono mechanizm pozwalający na proste blokowanie dostępu do systemu wszystkim użytkownikom poza użytkownikiem *root*. Polega on na umieszczeniu w katalogu */var/run* pliku o nazwie *nologin*. W najprostszym przypadku plik może być pusty, gdyż program logujący w pierwszej kolejności sprawdza jego istnienie. Jeśli natomiast plik posiada zawartość, to jest ona wypisywana na ekranie podczas podłączania się do systemu. Stanowi to wygodny

mechanizm informowania o przyczynach braku możliwości podłączenia się do systemu. Należy zaznaczyć, że utworzenie pliku `/var/run/nologin` w żaden sposób nie działa na użytkowników, którzy już są do systemu podłączeni. Jeśli nie chcemy ich obecności w systemie, to muszą oni zostać „ręcznie” odłączeni. Będąc użytkownikiem *root* możemy odłączyć od systemu każdego użytkownika kończąc działanie procesu, dzięki któremu jest on podłączony do systemu. Istnienie pustego pliku `/var/run/nologin` objawia się podczas podłączania do systemu następująco:

```

1 [bory@thorin ~]$ ssh wacek@141.152.99.13
2 Password:
3 pam_nologin: pam_sm_acct_mgmt: Administrator refusing you: /var/run/nologin
4
5
6 Permission denied (publickey,keyboard-interactive).
```

Zazwyczaj sytuacje, w których cała społeczność użytkowników systemu nie będzie miała do niego dostępu wynikają z upgradu oprogramowania lub systemu, bądź wykonywania kopii zapasowej lub backupu. Są zatem wcześniej planowane. Dobrym zwyczajem jest informowanie użytkowników o tego typu niedogodnościach odpowiednio wcześniej. Program logujący podczas podłączania użytkownika do systemu sprawdza, czy w katalogu `/etc` istnieje plik o nazwie *motd*. Jeśli tak, to jego zawartość jest wypisywana na ekranie. Mechanizm ten jest często wykorzystywany przez administratorów do informowania użytkowników o planowanych zmianach w działaniu systemu, usuniętych awariach czy zainstalowanym oprogramowaniu (patrz podrozdział 2.2.4).

### Plik `/etc/login.access`

Plik `/etc/login.access` pozwala na kontrolowanie możliwości podłączania się do systemu pojedynczego użytkownika lub grupy użytkowników z rozróżnieniem sposobu. Jego budowa oraz sposób interpretowania zapisanych w nim reguł podłączania się do systemu są identyczne jak dla pliku `/etc/security/access.conf` występującego w dystrybucjach Red Hat, a omówionego w podrozdziale 2.2.4.

### 2.6.5 Usuwanie użytkownika z systemu

Konto użytkownika może usunąć z systemu użytkownik *root*. Najprostszym sposobem jest użycie komendy *rmuser*. Zaczniemy od zapoznania się z czynnościami, jakie wykonuje komenda usuwając konto użytkownika. Są to:

1. Usunięcie pliku zawierającego specyfikację zadań do cyklicznego uruchamiania (z katalog `/var/cron/tabs`).
2. Usunięcie zadań użytkownika z kolejki zadań do uruchomienia (`/var/at/jobs`).
3. Wysłanie do wszystkich procesów użytkownika sygnału *SIGKILL*, będącego poleceniem ich natychmiastowego zakończenia.
4. Usunięcie definicji użytkownika z lokalnych plików konfiguracyjnych.
5. Usunięcie katalogu domowego użytkownika, jeśli jest on jego właścicielem jak również plików będące dowiązaniem symbolicznymi wskazującymi na usuwany katalog.
6. Usunięcie pliku z pocztą użytkownika (katalog `/var/mail`).

7. Usunięcie plików będących własnością użytkownika z katalogów `/tmp`, `/var/tmp`, `/var/tmp/vi.recover`.
8. Uporządkowanie pliku konfiguracyjnego grup użytkowników (`/etc/group`) polegające na usunięciu nazwy użytkownika z pliku definicji grup. Jeśli użytkownik był jedynym członkiem grupy o nazwie takiej jak nazwa użytkownika, to grupa ta zostaje również usunięta.
9. Usunięcie kolejek komunikatów procesów, segmentów pamięci współdzielonej oraz semaforów przydzielonych przez system procesom użytkownika.

Jak widać funkcjonalność komendy `rmuser` w systemie FreeBSD jest nieporównywalnie większa niż jej odpowiedniczki w rodzinie Red Hat. Można jej zarzucić jedynie brak możliwości archiwizowania plików usuwanego użytkownika.

Komenda pracuje zasadniczo w trybie interaktywnym. Uruchomienie jej bez opcji i argumentów powoduje, że prosi ona o podanie nazwy jednego lub kilku użytkowników, którzy mają zostać z systemu usunięci (linia 2 poniższego listingu). Następnie wypisuje linię definiującą użytkownika w pliku `/etc/passwd` (linia 5) i monituje o potwierdzenie (linia 6). Następnie pyta, czy usunąć katalog domowy użytkownika (linia 7), po czym informuje, że konto danego użytkownika zostało z systemu usunięte (linia 8).

```
1 # rmuser
2 Please enter one or more usernames: jan
3 Matching password entry:
4
5 jan:*:1002:1002::0:0:Jan Kowalski:/home/jan:/bin/sh
6
7 Is this the entry you wish to remove? y
8 Remove user's home directory (/home/jan)? y
9 Removing user (jan): mailspool home passwd.
```

Komendę można uruchomić z argumentem (lub argumentami) będącym nazwą użytkownika. Takie uruchomienie spowoduje, że komenda nie będzie wymagała podania go (ich) w trakcie pracy. Jednak w przypadku usuwania dużej liczby użytkowników wielokrotne potwierdzanie czynności klawiszem "y" może być stresujące. Z pomocą przychodzą opcje komendy:

- **-y** – użycie tej opcji powoduje automatyczne generowanie odpowiedzi twierdzących na każde pytanie zadawane przez komendę `rmuser`.
- **-f *nazwa\_pliku*** – plik, do którego ścieżkę dostępu podaje się jako wartość opcji jest plikiem tekstowym zawierającym, po jednej w linii, nazwy użytkowników, którzy mają zostać usunięci z systemu. Istotne jest, aby właścicielem indywidualnym pliku był użytkownik `root` oraz aby nikt poza właścicielem indywidualnym nie miał do niego prawa zapisu. Jeśli którykolwiek z warunków nie będzie spełniony, komenda `rmuser` nie wykona się.

Decydując się na pozostawienie katalogu domowego użytkownika należy być szczególnie ostrożnym przy przydzielaniu numerów identyfikacyjnych nowym użytkownikom. Przypomnijmy, że jądro stwierdza, że właścicielem pliku jest dany użytkownik, jeśli jego numer identyfikacyjny UID jest taki jak numer właściciela indywidualnego zapisany w strukturze opisującej plik. Stąd nowy użytkownik staje się właścicielem wszystkich plików usuniętego użytkownika, jeśli tylko posiada ten sam numer identyfikacyjny.

## 2.7 Zarządzanie grupami użytkowników w dystrybucji FreeBSD

### 2.7.1 Dodawanie grupy użytkowników

W dystrybucji FreeBSD grupę użytkowników możemy dodać do systemu edytując plik podstawowy zawierający definicje grup, a więc plik `/etc/group`. Dodanie grupy sprowadza się do dopisania linii składającej się z czterech pól oddzielonych znakiem ":", a zawierających kolejno nazwę grupy, \* oznaczającej nieistotność pola przechowującego postać zakodowaną hasła grupowego, numer identyfikacyjny oraz listę nazw użytkowników, którzy do tej grupy należą, ale nie jest to ich grupa podstawowa. Należy pamiętać aby numer grupy oraz jej nazwa były unikalne. Lista członków grupy może być początkowo pusta lub zawierać nazwy użytkowników nieistniejących w systemie. Widać, że taki sposób postępowania jest bardzo podatny na błędy, a zatem przeznaczony dla użytkowników posiadających pewne doświadczenie oraz dobrą znajomość systemu.

Zdefiniowanie grupy z zapewnieniem kontroli poprawności zapewni komenda `pw`, którą należy użyć ze słowami kluczowymi **group** oraz w tym wypadku **add**. Wymagany argumentem komendy jest nazwa grupy, która ma zostać dodana. Komenda sprawdzi, czy proponowana nazwa nie jest już wykorzystywana oraz nada pierwszy wolny numer identyfikacyjny. Wykonanie komendy:

```
1 # pw group add projekt
```

spowoduje, że w pliku `/etc/group` pojawi się linia:

```
1 projekt*:1004
```

Odpowiednie opcje pozwalają na modyfikowanie lub nadanie odpowiednich wartości atrybutom definiowanej grupy. Do najważniejszych należą następujące:

1. **-C** *plik\_konfiguracyjny* – opcja pozwala określić ścieżkę dostępu do pliku konfiguracyjnego komendy `pw`, zawierającego wartości domniemane atrybutów dla definiowanej grupy. Plikiem domniemanym jest plik `pw.conf` znajdujący się w katalogu `/etc`.
2. **-q** – użycie opcji blokuje wypisywanie komunikatów o błędach, co jest wykorzystywane zwłaszcza podczas wywoływania komendy w skryptach.
3. **-n** *nazwa\_grupy* – wartość opcji to nazwa definiowanej grupy.
4. **-g** *numer\_identyfikacyjny* – opcja pozwala na nadanie numeru identyfikacyjnego definiowanej grupie.
5. **-M** *lista\_członków\_grupy* – wartością opcji jest lista nazw lub numerów identyfikacyjnych oddzielonych przecinkami użytkowników zdefiniowanych w systemie. Dzięki opcji mamy możliwość zdefiniowania członków grupy już na etapie jej definiowania. Grupa ta będzie dla nich grupą dodatkową.

Dodanie do systemu grupy o nazwie *pracownicy*, numerze identyfikacyjnym *1025* oraz jednocześnie zdefiniowanie tej grupy jako dodatkowej dla użytkowników *wacek* oraz *antek* wymaga wydania komendy:

```
1 # pw groupadd -n pracownicy -g 1025 -M wacek,antek
```

W wyniku jej wykonania w pliku */etc/group* została dopisana następująca linia:

```
1 pracownicy::1025:wacek,antek
```

### 2.7.2 Zmiana konfiguracji grupy

Jak wspomniano, zmiana konfiguracji grupy użytkowników może mieć dwojaki charakter. W pierwszym wypadku może dotyczyć wartości podstawowych atrybutów grupy, a więc obejmować zmianę nazwy grupy, czy jej numeru identyfikacyjnego. Zwłaszcza zmiana numeru identyfikacyjnego grupy musi zostać przeprowadzona starannie i nie ograniczyć się jedynie do zmiany w trzecim polu linii definiującej grupę w pliku */etc/passwd*. Zmianie ulega wówczas także definicja grupy podstawowej użytkownika, a więc należy poprawić również zawartość plików konfiguracyjnych i bazy danych użytkowników. Dodatkowo należy pamiętać, że grupa podstawowa decyduje o właścicielu grupowym. Stąd należy także zmienić właściciela grupowego wszystkich plików, których dotychczasowym właścicielem była modyfikowana grupa.

Zmianę wartości atrybutów grupy może przeprowadzić użytkownik *root*. Najprostszy sposób polega na edycji pliku konfiguracyjnego */etc/group*. Komendą umożliwiającą wprowadzenie zmian bez znajomości jego budowy jest komenda *pw*. Należy ją użyć ze słowami kluczowymi **group** oraz **mod**. Następnie powinna pojawić się nazwa grupy, wartości atrybutów której będą modyfikowane. Do najczęściej wykorzystywanych opcji należą:

- **-g numer\_identyfikacyjny** – pozwala na zmianę numeru identyfikacyjnego na będący wartością opcji. Proponowany numer identyfikacyjny musi być unikalny.
- **-l nazwa\_grupy** – opcja umożliwia zmianę nazwy grupy na podaną jako wartość opcji. Grupa o nazwie proponowanej nie może już istnieć w systemie.
- **-M lista\_członków\_grupy** – wartością opcji jest lista nazw lub numerów identyfikacyjnych użytkowników oddzielonych przecinkami zdefiniowanych w systemie. Opcja pozwala na zdefiniowanie nowej listy użytkowników systemu będących członkami grupy.
- **-m lista\_użytkowników** – podobnie jak w przypadku opcji **-M** wartością opcji jest lista nazw lub numerów identyfikacyjnych użytkowników systemu oddzielonych przecinkami. Opcja służy do dodawania ich jako nowych członków grupy przy zachowaniu dotychczasowych.

Komenda *pw* dokonuje zmian w pliku konfiguracyjnym grup. Należy pamiętać, aby po zmianie atrybutów grupy dokonać modyfikacji zawartości bazy danych użytkowników używając komendy *pwd\_mkdb* z argumentem będącym ścieżką dostępu do pliku konfiguracyjnego grup.

Zdefiniowanej uprzednio grupie *pracownicy* o numerze identyfikacyjnym 1025 możemy zmienić numer używając komendy:

```
1 # pw groupmod pracownicy -g 1039
```

Usunięcie wszystkich członków grupy realizujemy komendą *pw* użytą z opcją **-M** z pustą listą użytkowników:

```
1 # pw groupmod pracownicy -M ""
```

Dodanie użytkownika *antek* do grupy przeprowadzi komenda:

```
1 # pw groupmod pracownicy -m antek
```

Po przeprowadzeniu powyższych czynności linia definiująca grupę *pracownicy* w pliku */etc/group* będzie miała postać:

```
1 pracownicy::1039:antek
```

### 2.7.3 Usuwanie grupy użytkowników

Czynność usuwania grupy użytkowników z systemu wykonuje się raczej sporadycznie. Zazwyczaj ma to miejsce przy okazji usuwania użytkownika z systemu w sytuacji, gdy był on jedynym członkiem grupy. Może być to jego grupa dodatkowa lub podstawowa. Usunięcie, podobnie jak modyfikację wartości atrybutów można przeprowadzić edytorem, usuwając linię definiującą grupę z pliku */etc/passwd* oraz, jeśli grupa ta była podstawową dla jakiegoś użytkownika, modyfikując jego atrybuty. Inny sposób polega na użyciu komendy *pw* ze słowami kluczowymi **group** oraz **del**. Komenda wymaga podania nazwy lub numeru identyfikacyjnego grupy, która ma zostać usunięta. Podajemy je jako argument wywołania komendy lub jako wartości opcji **-n** dla nazwy lub **-g** dla numeru identyfikacyjnego. Podobnie, jak w przypadku zmiany atrybutów grupy należy pamiętać o uaktualnieniu zawartości bazy danych użytkowników.

Niestety, działanie komendy *pw* w tym zakresie pozostawia wiele do życzenia. Przede wszystkim należy zwrócić uwagę na fakt, że komenda nie sprawdza, czy usuwana grupa jest podstawową dla użytkownika zdefiniowanego w systemie. Grupa taka jest usuwana bez żadnego komunikatu. Problemy pojawiają się dopiero wówczas, gdy zaistnieje konieczność ustalenia właściciela grupowego dla istniejących plików lub procesów lub tworzonych przez użytkownika, którego grupę podstawową właśnie usunięto. W pierwszym przypadku pojawi się numer identyfikacyjny zapisany w i-węzle danego pliku lub strukturze opisującej proces w tablicy procesów jądra systemu operacyjnego, w drugim zostanie wykorzystany numer grupy podstawowej użytkownika zapisany w bazie danych użytkowników, a pochodzący z pliku */etc/master.passwd*. Prześledźmy to na przykładzie. Usuwamy grupę podstawową użytkownika *antek* poleceniem:

```
1 # pw groupdel antek
```

Właścicielem pliki znajdujących się w katalogu domowym użytkownika *antek* jest nieistniejąca grupa o numerze 1003. Listowanie zawartości katalogu domowego w postaci długiej daje następujący wynik:

```
1 $ ls -al
2 total 22
3 drwxr-xr-x  2 antek  1003   512 Apr 26 14:39 .
4 drwxr-xr-x  5 root   wheel  512 Apr 20 13:31 ..
5 -rw-r--r--  1 antek  1003   751 Mar 16 14:02 .cshrc
6 -rw-r--r--  1 antek  1003   248 Mar 16 14:02 .login
7 -rw-r--r--  1 antek  1003   158 Mar 16 14:02 .login_conf
```

O niespójnościach w plikach konfiguracyjnych grup świadczy fakt pojawienia się numeru identyfikacyjnego właściciela grupowego, a nie jego nazwy. Konsekwencje są łatwe do przewidzenia. Wystarczy, że zdefiniujemy w systemie grupę o numerze identyfikacyjnym usuniętej grupy podstawowej, a stanie się ona ich nowym właścicielem. Stąd wszyscy jej członkowie będą mieli takie prawa dostępu do nich, jakie obowiązują właściciela grupowego. Przykładowo wejścia do katalogu domowego, odczytu jego zawartości oraz zawartości plików w nim znajdujących się.

Niedoskonałości narzędzi służących do usuwania grupy użytkowników w tej dystrybucji systemu sugerują napisanie prostego skryptu, który wykona następujące czynności:

1. Sprawdzi, czy usuwana grupa jest podstawową dla użytkownika systemu. Jeśli tak, to wypisze stosowny komunikat zawierający nazwę użytkownika i zakończy działanie. W ten sposób administrator będzie mógł dla znalezionego użytkownika zdefiniować nową grupę podstawową.
2. Znajdzie w systemie pliki, których właścicielem grupowym jest usuwana grupa. Dla każdego znalezionego pliku dokona zmiany właściciela grupowego na grupę, która została zdefiniowana jako nowa grupa podstawowa (komenda *chgrp*).
3. Usunie z pliku konfiguracyjnego grup linię definiującą usuwaną grupę.

Przykładowy skrypt zamieszczono w rozdziale 5.4.

## 2.8 Spójność plików konfiguracyjnych w dystrybucji FreeBSD

### 2.8.1 Pliki konfiguracyjne użytkowników

Przypomnijmy, że podstawowym plikiem konfiguracyjnym w dystrybucji FreeBSD jest plik *master.passwd* znajdujący się w katalogu *etc*. Plikiem dodatkowym jest plik */etc/passwd*. Został on zachowany dla zgodności z innymi dystrybucjami systemu. Jest tworzony z pliku */etc/master.passwd* przez usunięcie atrybutów mających istotne znaczenie dla bezpieczeństwa systemu, jak np. postać zaszyfrowana hasła. Trzecim plikiem przechowującym informacje o użytkownikach systemu jest plik bazy danych użytkowników */etc/pwd.db*. Jego zadaniem jest skracanie czasu dostępu do wartości atrybutów użytkowników przy ich liczbie sięgających kilka tysięcy. Zawartość pliku bazy danych jest tworzona również w oparciu o plik */etc/master.passwd*. Stąd zapewnienie poprawności danych w tym pliku załatwia problem spójności informacji o użytkownikach zdefiniowanych w systemie.

Zaproponowane rozwiązanie polega na wprowadzenia programu *vipw*, służącego do edycji pliku */etc/master.passwd*. Program ten działa w dwóch fazach:

1. Uruchomienie skutkuje uruchomieniem edytora (zdefiniowanego zmienną środowiskową *EDITOR*, w konfiguracji domyślnej jest to edytor *vi*) i wczytaniem pliku */etc/master.passwd*.
2. Po wprowadzeniu modyfikacji i wyjściu z trybu edycji z zapisaniem pliku, uruchamiany jest program *pwd\_mkdb*. Przed uaktualnieniem informacji w pozostałych plikach konfiguracyjnych sprawdzana jest poprawność konwertowanej informacji na podstawowym poziomie. Przykładowo, brak numeru identyfikacyjnego grupy podstawowej zostanie znaleziony, natomiast sprawdzanie, czy wpisany numer odpowiada zdefiniowanej grupie nie jest przeprowadzane. Jeśli zostaną wykryte błędy, pojawia się stosowny komunikat i pytanie o ewentualną, ponowną edycję.

```

1 pwd_mkdb: no gid for user antek
2 pwd_mkdb: at line #27
3 pwd_mkdb: /etc/pw.v9vvMJ: Inappropriate file type or format
4 re-edit the password file?

```

Jeśli w pliku */etc/master.passwd* nie znaleziono błędów, uaktualniana jest informacja w pozostałych plikach konfiguracyjnych i program *vipw* kończy działanie.

Zatem komendą umożliwiającą sprawdzanie poprawności plików konfiguracyjnych użytkowników jest *pwd\_mkdb*. Należy ją uruchamiać zawsze, jeśli modyfikacji pliku */etc/master.passwd* dokonywaliśmy edytorem bez korzystania z komendy *vipw*.

### 2.8.2 Pliki konfiguracyjne grup

Polecenie *chkgrp* służy sprawdzaniu poprawności pliku definiującego grupy w systemie FreeBSD. Uruchomiona bez argumentu pracuje na zawartości pliku */etc/group*. Argumentem może być ścieżka dostępu do pliku przechowującego definicję grup, którego poprawność ma zostać sprawdzona. Weryfikacja polega na sprawdzeniu:

- Czy każda niepusta linia, która nie jest komentarzem jest zbudowana z czterech kolumn oddzielonych ":".
- Czy w żadnym polu linii definiującej grupę nie pojawiły się białe znaki.
- Czy w trzeciej kolumnie znajduje się liczba (będąca numerem identyfikacyjnym grupy).
- Sprawdzane jest także, czy nazwa grupy jak również nazwy użytkowników będących członkami grupy są poprawne, w szczególności czy nie zawierają znaków specjalnych lub nie są w postaci numerycznej.

Komenda *chkgrp* raportuje błędy w postaci nazwy sprawdzanego pliku, numeru linii, w której błąd został znaleziony oraz krótkiego opisu błędu. Przykładowo:

```

1 # chkgrp
2 chkgrp: /etc/group: line 34: missing field(s)
3 chkgrp: /etc/group: line 35: ' ' invalid character
4 chkgrp: /etc/group: line 35: '#' invalid character
5 chkgrp: /etc/group: line 35: field 1 contains whitespace

```

Linia 1 powyższego listingu informuje o brakującym polu w linii 34 pliku konfiguracyjnego */etc/group*. W linii 35 pojawiła się spacja oraz znak # (linie 2 oraz 3). Linia 4 listingu zawiera powtórzenie informacji z linii 2. Błędy muszą zostać usunięte „ręcznie”, gdyż komenda jedynie o nich informuje. Na poprawny składniowo plik */etc/group* komenda *chkgrp* reaguje komunikatem:

```

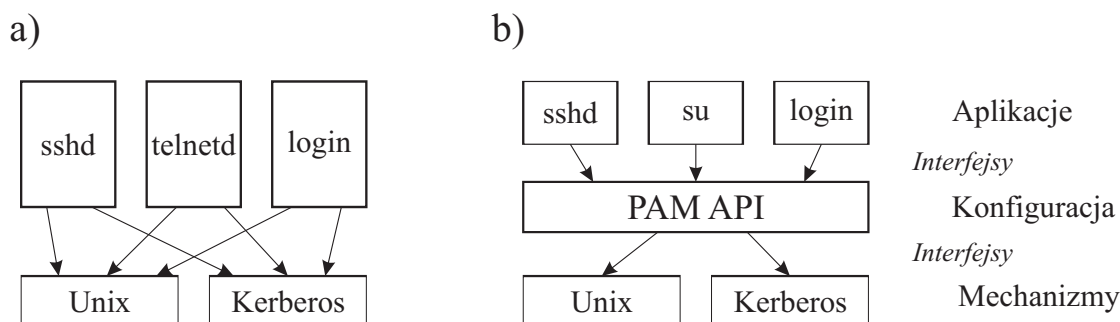
1 # chkgrp
2 /etc/group is fine

```



## 2.9 Pluggable Authentication Modules (PAM)

Jednym z zadań administratora systemu wieloużytkownikowego jest zapewnienie bezpieczeństwa przechowywanych w nim danych bez konieczności nakładania dodatkowych ograniczeń na użytkowników. W przypadku procesu autentykacji użytkowników w systemie ograniczenia te objawiają się przede wszystkim jego skomplikowaniem. Z kolei coraz nowsze usługi wymagają przechowywania i kontrolowania dodatkowych informacji o użytkowniku. Stąd tradycyjny system oparty pierwotnie na pliku `/etc/passwd` lub `/etc/master.passwd`, a następnie dodatkowo na plikach `/etc/shadow` lub bazodanowym pliku `/etc/pwd.db` szybko stał się ograniczony. Jego wadami są przede wszystkim sztywny format plików konfiguracyjnych oraz brak możliwości centralizacji lub rozproszenia danych użytkowników w dużych systemach informatycznych. W takim rozwiązaniu jakiejkolwiek zmiany formatu plików pociągają za sobą modyfikację kodu źródłowego aplikacji, które z nich korzystały. Schematycznie przedstawiono to na rysunku 2.1a.



Rysunek 2.1: Architektura PAM.

Występowanie istotnych ograniczeń w tradycyjnym sposobie uwierzytelniania oraz brak możliwości elastycznej konfiguracji z jednej strony, a coraz większe wymagania usług i aplikacji z drugiej doprowadziły do powstania projektu *Pluggable Authentication Modules (PAM)*, który w języku polskim możemy nazwać modularnym systemem uwierzytelniania. Pierwsza implementacja pojawiła w roku 1995 w systemach firmy SUN. Pomysł upowszechnił się. Pojawiło się wiele jego implementacji, co wymusiło powstanie specyfikacji *X/Open Single Sign-on (XSSO)* opublikowanej przez *Open Group*, definiującej standard PAM. Niestety sam standard nie uzyskał szerszej akceptacji, stąd istniejące implementacje PAM różnią się między sobą.

Idea modularnego systemu uwierzytelniania została przedstawiona na rysunku 2.1b. Polega ona na przeniesieniu ciężaru autentykacji na dołączane moduły, które mogą być projektowane i implementowane przez użytkowników. Stąd zmiana metody autentykacji z tradycyjnej opartej na nazwie użytkownika i skojarzonym z nią hasłem na dowolną metodę biometryczną wymaga stworzenia modułu, który będzie komunikował się z urządzeniem służącym do autentykacji oraz, dzięki zdefiniowanym interfejsom z programem, który w systemie operacyjnym jest wykorzystywany do autentykacji. W razie zmiany urządzenie zewnętrzne, w rozwiązaniu tym zmieniamy jedynie moduł, a nie aplikację.

### 2.9.1 Konfiguracja

System PAM składa się z dwóch elementów. Są to:

1. tekstowe pliki konfiguracyjne, które zawierają opis sposobu użycia modułów dla każdej aplikacji wymagającej uwierzytelnienia. Znajdują się one zazwyczaj w katalogu `/etc/pam.d`, a

ich nazwa odpowiada zazwyczaj nazwie aplikacji. W rozwiązaniu stosowanym w systemie Solaris konfiguracja jest przechowywana w pliku */etc/pam.conf*.

2. moduły wykorzystywanych przez aplikacje, które zazwyczaj znajdują się w katalogu */lib/security* (lub */lib64/security*). Są to biblioteki dołączane dynamicznie.

Ogólny schemat aplikacji wykorzystującej PAM jest następujący:

```

1  #include <security/pam_appl.h>
2  #include <security/pam_misc.h>
3
4  int main (int argc, char **argv){
5
6      pam_handle_t *pamh = NULL;
7      int retval;
8      const char *user = "nobody";
9      //
10     // Skopiowanie nazwy uzytkownika jako pierwszego argumentu wywołania programu
11     //
12     if ( argc == 2 )
13         user = argv[1];
14     else
15         exit (1);
16     //
17     // Inicjalizacja PAM
18     //
19     retval = pam_start ("chk_user", user, &conv, &pamh);
20     //
21     // Jesli inicjalizacja się powiodła, dokonujemy autentykacji uzytkownika
22     //
23     if ( retval == PAM_SUCCESS ) retval = pam_authenticate (pamh, 0);
24     //
25     // Jesli autentykacja powiodła się sprawdzamy, czy spelnione sa
26     // kryteria logowania (np. godziny)
27     //
28     if ( retval == PAM_SUCCESS ) retval = pam_acct_mgmt (pamh, 0);
29     //
30     // Konczymy prace z systemem PAM
31     //
32     if ( pam_end (pamh, retval) != PAM_SUCCESS ) {
33         pamh = NULL;
34         exit (1);
35     }
36     //
37     return ( retval == PAM_SUCCESS ? 0 : 1 );
38 }

```

Aplikacja przeprowadza autentykację użytkownika (funkcja *pam\_authenticate*) oraz stan jego konta (funkcja *pam\_acct\_mgmt*) w sensie, czy nie zostało ono zablokowane lub czy godziny podłączania się do systemu są właściwe. Gdzie występuje połączenie między składnikami modułu PAM,

a powyższym programem? W wywołaniu funkcji *pam\_start*, inicjalizującej system, pierwszym argumentem jest napis *chk\_user*. Jeśli konfiguracja PAM zapisana jest w pliku */etc/pam.conf* to każda linia konfiguracyjna rozpoczynająca się od takiego napisu, konfiguruje tę aplikację. Dla ułatwienia przyjęto, że napis jest nazwą aplikacji. Konfiguracja oparta o katalog */etc/pam.d* charakteryzuje się tym, że poszczególne aplikacje mają własne pliki konfiguracyjne o nazwach takich jak pierwszy argument wywołania funkcji *pam\_start*. Zazwyczaj nazwy plików pokrywają się z nazwami aplikacji. W obu przypadkach nazwy aplikacji zapisuje się małymi literami. Przykładowo plik */etc/pam.d/login* zawiera konfigurację dla programu *login*. W systemie Fedora jest ona następująca:

```

1  #%PAM-1.0
2  auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so
3  auth      include      system-auth
4  account   required      pam_nologin.so
5  account   include      system-auth
6  password  include      system-auth
7  # pam_selinux.so close should be the first session rule
8  session   required      pam_selinux.so close
9  session   required      pam_loginuid.so
10 session   optional      pam_console.so
11 # pam_selinux.so open should only be followed by sessions to be executed in the user context
12 session   required      pam_selinux.so open
13 session   required      pam_namespace.so
14 session   optional      pam_keyinit.so force revoke
15 session   include      system-auth
16 session   optional      pam_ck_connector.so

```

Jak widać plik konfiguracyjny aplikacji jest plikiem tekstowym o budowie liniowej. Może zawierać komentarz, którym jest tekst występujący po znaku *#* do końca linii. Linie zawierają opis sposobu wywołania modułu dla obsługi danej aplikacji oraz sposobu postępowania zależnego od sposobu zakończenia jego pracy. Wykonywane są w kolejności występowania w pliku chyba, że zawierają instrukcję skoku. Mówi się, że tworzą one sekwencję lub stos wywołań. Każda linia może składać się z pół oddzielonych znakiem białym. Ogólna składnia linii jest następująca:

*nazwa\_usługi rodzaj\_zadania znacznik uruchamiany\_moduł argumenty\_modułu*

Dla konfiguracji zapisywanej w plikach w katalogu */etc/pam.d* pierwsze pole jest pomijane, gdyż każda usługa ma swój plik konfiguracyjny. Pojawia się ono jedynie w pliku */etc/pam.conf* dla odróżnienia w pliku konfiguracji poszczególnych usług.

Kolejne pole zawiera rodzaj (ang. *type*) zadania. System PAM spełnia cztery, a są to:

1. **auth** – autentykacja polegająca na sprawdzeniu nazwy użytkownika w systemie oraz hasła i na tej podstawie przydzielenie lub odrzucenie dostępu.
2. **account** – zarządzanie kontem, a w tym obsługą haseł, które utaciły ważność, obsługą zablokowanego konta oraz wszelkich limitów, które zostały na konto nałożone.
3. **password** – ogólne zarządzanie hasłem. Zadanie jest wykorzystywane np. w momencie zmiany hasła przez użytkownika.
4. **session** – zarządzanie sesją wykorzystywane we wszelkich czynnościach organizacyjnych podejmowanych przed uzyskaniem przez użytkownika dostępu do usługi, np. montowanie systemów plików, nadawanie wartości zmiennym środowiskowym.

Zawartość kolejnego pola stanowi znacznik kontrolny. Jest on używany do określenia, w jaki sposób system PAM ma zareagować na sukces lub niepowodzenie konkretnego modułu. Zdefiniowane są następujące flagi: *required*, *requisite*, *sufficient*, *optional*, *include* oraz *substack*. Ten sposób definiowania znaczników uważany jest za historyczny. Jest jednak powszechnie używany. Znaczniki są interpretowane w następujący sposób:

- **required** – sukces modułu jest wymagany do powodzenia grupy modułów danego typu. Informacja o niepowodzeniu zostanie przekazana do aplikacji użytkownika dopiero po wykonaniu wszystkich pozostałych modułów tego typu.
- **requisite** – działa podobnie jak znacznik *required*, ale w przypadku niepowodzenia informacja jest zwracana do aplikacji natychmiast.
- **sufficient** – zakończenie wykonywania bieżącego modułu wystarcza aby uznać za zakończone sukcesem wszystkie moduły tego typu, ale tylko wówczas gdy każdy z poprzednich modułów zakończył się sukcesem.
- **optional** – moduł oznaczony znacznikiem *optional* nie jest uważany za krytyczny dla systemu. PAM ignoruje taki moduł w momencie określania czy działanie całej grupy modułów tego typu należy uznać za zakończone sukcesem czy niepowodzeniem. Wyjątkiem jest przypadek, w którym pozostałe moduły nie zwróciły wartości ani sukcesu ani niepowodzenia lecz np. PAM\_IGNORE. W takiej sytuacji moduł ten jest wykorzystywany do określenia sumarycznego statusu wykonania grupy modułów.
- **include** – powoduje uwzględnienie wszystkich linii opisujących ten sam rodzaj zadania pochodzących z pliku, którego nazwa pojawiła się jako argument znacznika. Przykładowo linia:

1	account	include	system-auth
---	---------	---------	-------------

wymaga uwzględnienia wszystkich linii typu *account* z pliku *system-auth*. Jeśli wartościowanie dowolnej linii zostało określone przez *done* lub *die* kolejne linie nie będą wartościowane.

- **substack** – uwzględnij wszystkie linie opisujące ten sam rodzaj zadania pochodzące z pliku, którego nazwa pojawiła się jako argument znacznika. Jeśli wartościowanie dowolnej linii zostanie określone jako *done* lub *die*, to kolejne linie będą wartościowane.

Znacznik kontrolny może zostać zapisany w nieco bardziej złożonej, nowej formie. Jej składnia jest wówczas następująca:

1	[ Wartość1=Akcja1 Wartość2=Akcja2 ..... ]
---	---

Napis *WartośćN* oznacza wartość zwróconą przez moduł wywołany w linii, w której pojawiła się specyfikacja danego znacznika kontrolnego. Może on przyjąć wartość spośród następujących: *success*, *open\_err*, *symbol\_err*, *service\_err*, *system\_err*, *buf\_err*, *perm\_denied*, *auth\_err*, *cred\_insufficient*, *authinfo\_unavail*, *user\_unknown*, *maxtries*, *new\_authtok\_reqd*, *acct\_expired*, *session\_err*, *cred\_unavail*, *cred\_expired*, *cred\_err*, *no\_module\_data*, *conv\_err*, *authtok\_err*, *authtok\_recover\_err*, *authtok\_lock\_busy*, *authtok\_disable\_aging*, *try\_again*, *ignore*, *abort*, *authtok\_expired*, *module\_unknown*, *bad\_item*, *conv\_again*, *incomplete* oraz *default*. Ta ostatnia oznacza dowolną spośród pozostałych. Pełna lista wartości zwracanych przez moduł PAM znajduje się w pliku */usr/include/security/\_pam\_types.h*.

Jako *AkcjaN* może pojawić się liczba całkowita, która oznacza o ile linii należy przeskoczyć w wartościowaniu, czyli ile kolejnych wywołań modułów pominąć. Umożliwia to tworzenie ścieżek wykonań. Może także pojawić się jeden z następujących napisów:

- **ignore** – jeśli wartość ta została użyta w sekwencji (stosie) linii zawierających wywołania modułów to oznacza, że wartość zwrócona przez bieżący moduł ma nie zostać uwzględniona w wartościowaniu całej sekwencji.
- **bad** – oznacza, że wartość zwrócona przez wywołany moduł PAM może posłużyć do określenia przyczyny zakończenia wykonywania modułu niepowodzeniem. Jeśli wykonanie modułu bieżącego zakończyło się niepowodzeniem jako pierwszego w sekwencji, to wartość przez niego zwrócona zostaje przyjęta jako wartość wykonania sekwencji. Wykonywane są wszystkie moduły z sekwencji.
- **die** – działanie podobne do wartości *bad* z tą różnicą, że jeśli moduł zakończył się niepowodzeniem, to nie są wykonywane następne moduły z sekwencji.
- **ok** – znacznik mówi, że wartość zwrócona przez bieżący moduł ma zostać użyta jako wartość zwrócona przez całą sekwencję modułów. Wykonywana jest cała sekwencja modułów.
- **done** – działanie podobne do wartości *ok* z tą różnicą, że wykonywanie sekwencji modułów kończy się na bieżącym module, a wartość zwrócona do aplikacji jest wartością zwróconą przez ten moduł.
- **reset** – wyzeruj stos wartości zwróconych przez poprzednio wykonane moduły z bieżącym włącznie i przejdź do wykonania kolejnego modułu.

Między starą i nową notacją istnieją następujące odpowiedniki:

- **required** odpowiada [*success = oknew\_authtok\_reqd = okignore = ignoredefault = bad*]
- **requisite** odpowiada [*success = oknew\_authtok\_reqd = okignore = ignoredefault = die*]
- **sufficient** odpowiada [*success = oknew\_authtok\_reqd = donedefault = ignore*]
- **optional** odpowiada [*success = oknew\_authtok\_reqd = okdefault = ignore*]

Kolejne pole linii zawiera ścieżkę dostępu do dynamicznie ładowanego modułu. Przyjęto założenie, że jeśli nie jest to ścieżka bezwzględna (zaczyna się od znaku różnego od /) to moduły są poszukiwane w katalogu */lib/security*.

Ostatnie pole to lista argumentów przekazywanych do modułu w momencie jego wywołania.

## 2.9.2 Podstawowe moduły

### Moduły systemu PAM-Linux

Podstawowe moduły systemu PAM-Linux oraz ich krótki opis zamieszczono w tabeli 2.2. Dokładny opis stanowiłby oddzielny tom. Stąd zainteresowanych odsyłam do opracowania *The Linux-PAM System Administrators' Guide* autorstwa A.G. Morgana oraz T. Kukuka. Jest ono dostępne w postaci elektronicznej jako plik w formacie *pdf*, a istotną zaletą jest jego aktualność wynikająca z dbałości autorów.

Tablica 2.2: Podstawowe moduły systemu PAM-Linux.

Moduł	Opis
<i>pam_access</i>	Podstawowy moduł zarządzający dostępem do systemów lokalnych i zdalnych. Wykorzystuje informacje z pliku <i>/etc/security/access.conf</i> .
<i>pam_cracklib</i>	Moduł realizuje pobranie hasła oraz sprawdzenie jego poprawności w stosunku do zdefiniowania przy pomocy tegoż modułu reguł opisujących jego złożoność.
<i>pam_debug</i>	Moduł służący monitorowaniu wykorzystania stosu PAM.
<i>pam_deny</i>	Moduł ten jest używany do blokowania dostępu do systemu. Do aplikacji zwraca zawsze niepowodzenie.
<i>pam_echo</i>	Wykorzystywany do wypisywania komunikatu tekstowego informującego użytkownika o pracy systemu PAM.
<i>pam_env</i>	Umożliwia definiowanie oraz usuwanie zmiennych środowiskowych.
<i>pam_exec</i>	Moduł służy uruchamianiu zewnętrznego polecenia.
<i>pam_faildelay</i>	Umożliwia ustawienie czasu opóźnienia do uruchomienia kolejnej aplikacji, jeśli bieżąca zakończyła się porażką. Przykładowo z jakim opóźnieniem ma pojawić się monit o podanie nazwy użytkownika, jeśli poprzedniemu nie udało się podłączyć.
<i>pam_filter</i>	Moduł ma umożliwiać dostęp do komunikatów przesyłanych strumieniami wejściowym i wyjściowym przez użytkownika do i z aplikacji. Wykorzystywany do aplikacji terminalowych.
<i>pam_ftp</i>	Jest wykorzystywany do realizacji dostępu do serwera ftp dla użytkowników anonimowych ( <i>anonymous</i> ).
<i>pam_group</i>	Nie służy autentykacji użytkownika, a nadaniu mu członkostwa w grupie. Bazuje ono na usłudze dla której zostało nadane.
<i>pam_issue</i>	Moduł umożliwia zdefiniowanie znaku zachęty użytkownika według szablonu opisanego w pliku będącym argumentem wywołania modułu.
<i>pam_keyinit</i>	Moduł umożliwia sprawdzenie, czy wywołujący go proces ma klucz sesji identyczny z domniemanym kluczem sesji. Wykorzystywany także do generowania indywidualnych kluczy sesji.
<i>pam_lastlog</i>	Wypisuje informację o czasie ostatniego podłączenia użytkownika do systemu. Korzysta z pliku <i>/var/log/lastlog</i> .
<i>pam_limits</i>	Moduł umożliwia ograniczanie zasobów systemu komputerowego przydzielanych użytkownikom. Wykorzystuje informacje zapisane w pliku <i>/etc/security/limits.conf</i> .
<i>pam_listfile</i>	Udostępnia prosty mechanizm blokujący lub udostępniający usługi, wykorzystujący pliki konfiguracyjne. Przykładowo możemy zabronić korzystania z usługi <i>ftp</i> użytkownikom, których nazwy zostały zapisane w pliku.
<i>pam_localuser</i>	Umożliwia implementację kontroli podłączania się użytkowników do systemu.
<i>pam_mail</i>	Moduł implementuje usługę powiadamiania użytkownika o otrzymaniu nowego listu.
<i>pam_mkhome</i>	Działanie modułu polega na utworzeniu katalogu domowego użytkownikowi, który podłącza się do systemu, a w momencie rozpoczęcia sesji katalog ten nie istnieje. Do utworzonego katalogu kopiowane są pliki konfiguracyjne.
<i>pam_motd</i>	Moduł służy wypisywaniu komunikatu z pliku <i>/etc/motd</i> podczas otwierania sesji użytkownika.

Tablica 2.2: Podstawowe moduły systemu PAM-Linux cd.

Moduł	Opis
<i>pam_namespace</i>	Moduł umożliwia ustawienie indywidualnej przestrzeni nazw (ang. <i>namespace</i> ) w sesji użytkownika.
<i>pam_nologin</i>	Uniemożliwia połączenie się do systemu użytkowników, jeśli istnieje plik <i>/etc/nologin</i> .
<i>pam_permit</i>	To moduł, który zawsze umożliwia dostęp i nie wykonuje innych czynności.
<i>pam_rhosts</i>	Moduł umożliwia prosty sposób autentykacji przy dostępie zdalnym poprzez użycie tradycyjnych implementacji <i>rlogin</i> czy <i>rsh</i> .
<i>pam_rootok</i>	Autentykuje użytkownika jeśli jego numer identyfikacyjny (UID) wynosi 0.
<i>pam_securetty</i>	Moduł umożliwia połączenie się do systemu użytkownikowi <i>root</i> jeśli sesja przebiega z terminala, który został zdefiniowany jako bezpieczny (plik <i>/etc/securetty</i> ).
<i>pam_selinux</i>	Umożliwia zdefiniowanie kontekstu bezpieczeństwa dla procesu bieżącego interpretera poleceń.
<i>pam_shells</i>	Moduł weryfikuje, czy interpreter logującego użytkownika jest zdefiniowany w pliku <i>/etc/shells</i> .
<i>pam_succeeded_if</i>	Jest wykorzystywany do podejmowania decyzji czy autentykacja użytkownika zakończyła się niepowodzeniem czy sukcesem zależnie od wartości atrybutów konta użytkownika. Przykładowo umożliwia zmianę kontekstu na użytkownika <i>root</i> jeśli użytkownik bieżący należy do odpowiedniej grupy użytkowników.
<i>pam_tally</i>	Moduł zarządza licznikami połączeń użytkownika do systemu, które zakończyły się niepowodzeniem lub sukcesem i umożliwia na tej podstawie prowadzenie odpowiedniej polityki bezpieczeństwa.
<i>pam_time</i>	Umożliwia kontrolowanie czasu podłączania się użytkowników do systemu jak również terminali, z których użytkownicy mogą do systemu się podłączać.
<i>pam_umask</i>	Moduł jest wykorzystywany do nadawania wartości umaski decydującej o prawach dostępu do nowotworzonych plików i katalogów.
<i>pam_unix</i>	Podstawowy moduł służący autentykacji użytkownika, odczytowi i zmianie wartości atrybutów konta użytkownika.
<i>pam_userdb</i>	Służy do weryfikacji nazwy i hasła użytkownika przechowywanych w bazie danych użytkowników w formacie zgodnym z Berkeley DB.
<i>pam_warn</i>	Moduł wykorzystywany do zapisu informacji, której źródłem jest usługa, terminal, użytkownik lokalny lub sieciowy do plików dziennika systemowego.
<i>pam_wheel</i>	Podstawowym zadaniem modułu jest umożliwianie podłączania się do systemu, np. przez zmianę kontekstu, użytkownikom należącym do grupy <i>wheel</i> . Ogłonie wykorzystywany do zarządzania uprawnieniami wynikającymi z przynależności do grupy <i>wheel</i> .
<i>pam_xauth</i>	Służy zarządzaniu prawami użytkowników przy podłączaniu się do systemu poprzez zmianę kontekstu w trakcie pracy z interfejsem okienkowym.

### Moduły systemu PAM w dystrybucji BSD

Implementacje systemu PAM w dystrybucjach FreeBSD różnią się nieznacznie od tych zaimplementowanych w systemach linuksowych. W tabeli 2.3 zamieszczono opisy podstawowych z nich.

Tablica 2.3: Podstawowe moduły systemu PAM w systemach BSD.

Moduł	Opis
<i>pam_deny</i>	Moduł służy do uniemożliwienia dostępu do usługi. Zazwyczaj pojawia się jako ostatni w sekwencji modułów ze znacznikiem <i>sufficient</i> .
<i>pam_echo</i>	Moduł umożliwia wypisywanie komunikatów. Często wykorzystywany do śledzenia wykonania (debuggingu).
<i>pam_exec</i>	Umożliwia uruchamianie komend zewnętrznych. Wykorzystywany przykładowo do montowania katalogu domowego w czasie podłączania użytkownika do systemu, jeśli katalog ten znajduje się na innym hoście.
<i>pam_ftpusers</i>	Moduł jest wykorzystywany do zarządzania dostępem do usługi ftp. Umożliwia dostęp użytkownikom zdefiniowanym w pliku <i>/etc/ftpusers</i> .
<i>pam_group</i>	Służy do kontrolowania dostępności usług dla grup użytkowników. Jest wykorzystywany przede wszystkim do możliwości komendy <i>su</i> , umożliwiając przełączenie się na konto użytkownika <i>root</i> jedynie członkom grupy <i>wheel</i> .
<i>pam_guest</i>	Służy do autentykacji użytkowników o niskich uprawnieniach w systemie (użytkownik <i>gość</i> ) i ustalonych nazwach. Częste zastosowanie to obsługa użytkowników anonimowych ( <i>anonymous</i> ) korzystających z serwera ftp.
<i>pam_krb5</i>	Moduł integrujący system PAM z mechanizmami autentykacji wykorzystywanymi w systemie Kerberos.
<i>pam_ksu</i>	Moduł umożliwia autentykację dla podłączania się do systemu przez zmianę kontekstu (komendą <i>su</i> ) z wykorzystaniem systemu Kerberos.
<i>pam_lastlog</i>	Służy dziennikowaniu podłączania się użytkowników do systemu. Wpisy pojawiają się w pliku <i>/var/log/lastlog</i> .
<i>pam_login_access</i>	Umożliwia nakładanie na użytkowników ograniczeń zdefiniowanych w pliku <i>/etc/login.access</i> .
<i>pam_nologin</i>	Moduł ten uniemożliwia podłączanie się do systemu użytkownikom, jeśli został utworzony plik <i>/var/run/nologin</i> . Restrykcja nie dotyczy użytkownika <i>root</i> .
<i>pam_opie</i>	Moduł implementuje metodę autentykacji <i>opie</i> . Metoda ta wykorzystuje własną funkcję skrótu do kodowania pakietów oraz system wymiany pakietów informacji zwany <i>challenge/response</i> .
<i>pam_opieaccess</i>	Moduł ten stanowi funkcjonalne uzupełnienie modułu <i>pam_opie</i> . Działa w oparciu o zawartość pliku <i>/etc/opieaccess</i> , który określa warunki dla autentykacji systemem <i>opie</i> lub innym. Moduł jest często wykorzystywany do autentykacji użytkowników podłączających się do systemu z hostów, które nie są zaufanymi.
<i>pam_passwdqc</i>	Moduł wykorzystywany do sprawdzania jakości haseł.
<i>pam_permit</i>	Moduł, jako kod powrotu każdego wywołania zwraca sukces.
<i>pam_radius</i>	Umożliwia wykorzystanie protokołu RADIUS (Remote Authentication Dial In User Service) do autentykacji użytkowników w systemie PAM.
<i>pam_rhosts</i>	Moduł jest wykorzystywany do autentykacji użytkowników podłączających się zdalnie. Zwraca sukces, jeśli numer identyfikacyjny użytkownika jest większy od 0 oraz jeśli podłącza się on do systemu z hosta, który został zdefiniowany w pliku <i>/etc/hosts.equiv</i> lub w pliku <i>.rhosts</i> w katalogu domowym użytkownika.



Tablica 2.3: Podstawowe moduły systemu PAM w systemach BSD  
cd.

Moduł	Opis
<i>pam_rootok</i>	Wykonanie modułu kończy się sukcesem, jeśli efektywny numer identyfikacyjny autentykowanego użytkownika wynosi 0. Wykorzystywany w komendach <i>su</i> czy <i>passwd</i> , które wywołane przez użytkownika <i>root</i> nie pytają o hasło.
<i>pam_securetty</i>	Moduł umożliwia podłączenie się do systemu użytkownikowi <i>root</i> jeśli sesja przebiega z terminala, który został zdefiniowany jako bezpieczny (zdefiniowany w pliku <i>/etc/ttys</i> ).
<i>pam_self</i>	Wykonanie modułu kończy się sukcesem, jeśli nazwa użytkownika, który podłącza się do systemu jest identyczna z nazwą użytkownika bieżącego. Moduł wykorzystywany do autentykacji połączeń lokalnych przez zmianę kontekstu (komendą <i>su</i> ).
<i>pam_ssh</i>	Moduł wykorzystywany do autentykacji oraz kontroli sesji. W pierwszym zastosowaniu umożliwia zdalne podłączenie do systemu z wykorzystaniem kluczy, jeśli użytkownik posiada je w katalogu <i>.ssh</i> znajdującym się w katalogu domowym. W sesji moduł uruchamia agenta <i>ssh</i> ( <i>ssh-agent</i> to program, który przechowuje klucze prywatne wykorzystywane do autentykacji kluczy prywatnych) i dokonuje uaktualnienia kluczy utworzonych podczas autentykacji użytkownika. Wykorzystywany w połączeniach lokalnych terminalowych i w systemach X.
<i>pam_tacplus</i>	Moduł wykorzystywany do autentykacji użytkowników z wykorzystaniem protokołu <i>TACACS+</i> (Terminal Access Controller Access-Control System Plus).
<i>pam_unix</i>	Moduł ten implementuje klasyczny, oparty o hasło, sposób autentykacji. W systemie Unix, wykorzystując funkcję <i>getpwnam</i> do pobrania postaci zaszyfrowanej hasła z bazy danych użytkowników i porównanie jej z zaszyfrowanym hasłem podanym przez użytkownika. Moduł obsługuje również kwestie związane z upływaniem daty ważności hasła i konta oraz sposobu zmiany hasła.

### 2.9.3 Przykłady zastosowań

#### Lista historii haseł

Modulem odpowiedzialnym jest moduł *pam\_unix*. Konfiguracja zapisana jest w pliku */etc/pam.d/system-auth*. Linia odpowiedzialna wygląda następująco:

```
1 password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
2 use_authtok
```

Jedyna modyfikacja polega na dodaniu flagi *remember* z wartością równą liczbie pamiętanych, ostatnio wykorzystywanych haseł. W przypadku listy o długości 3 linia ma następującą postać:

```
1 password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
2 use_authtok remember=3
```

Lista użytych haseł będzie pamiętana w pliku tekstowym o nazwie *opasswd* znajdującym się w katalogu */etc/security*. Jeśli plik ten nie istnieje, to należy go utworzyć, zmienić właściciela

indywidualnego i grupowego odpowiednio na użytkownika *root* oraz grupę *root* oraz zmienić prawa dostępu tak, aby jedynie właściciel indywidualny miał prawo odczytu i zapisu. Czynności te wykonujemy będąc użytkownikiem *root*, np. przy pomocy następującego ciągu komend:

```
1 [root@messy pam.d]# touch /etc/security/opasswd
2 [root@messy pam.d]# chown root:root /etc/security/opasswd
3 [root@messy pam.d]# chmod 600 /etc/security/opasswd
4 [root@messy pam.d]#
```

Próba użycia hasła, które znajduje się na liście ostatnio wykorzystywanych wygląda następująco:

```
1 [antek@messy ~]$ passwd
2 Changing password for user antek.
3 Changing password for antek.
4 (current) UNIX password:
5 New UNIX password:
6 BAD PASSWORD: has been already used
7 New UNIX password:
```

Plik */etc/security/opasswd* jest plikiem tekstowym o budowie linikowej przedstawione poniżej:

```
1 antek:502:2:$1$BiFvb/6T$BwS3J/U16F0XLaW8Wf997. , $1$1mbdTw.5$5fHpZ815L7xExE.LSNYLr1
```

Jedna linijka opisuje jednego użytkownika i składa się z czterech kolumn oddzielonych znakiem ":". W kolumnie pierwszej przechowywana jest nazwa użytkownika w systemie. Kolumna druga zawiera jego numer identyfikacyjny. W kolumnie trzeciej znajduje się liczba mówiąca ile ostatnio używanych przez użytkownika haseł znajduje się na liście. W naszym przypadku zapamiętane zostały tylko 2 hasła. Pole czwarte to lista haseł w postaci zaszyfrowanej, oddzielonych przecinkami.

Stosując opisywany mechanizm w powiązaniu z minimalnym okresem ważności hasła możemy mieć pewność, że „ulubione” hasło użytkownika nie będzie używane zbyt często. Jak wspomniano, liczba dni, przez które hasło nie może zostać użyte, to iloczyn minimalnego okresu ważności hasła i długości listy używanych haseł.

### Nieudane próby połączenia do systemu

Próby połączenia do systemu zakończone niepowodzeniem mogą być wynikiem próby odgadnięcia hasła przez potencjalnego włamywacza. Stąd istotna dla bezpieczeństwa systemu jest możliwość zablokowania dostępu do systemu użytkownikowi, który kilka następujących po sobie razy podał błędne hasło. Modułem umożliwiającym śledzenie nieudanych prób połączenia do systemu jest moduł *pam-tally*. Konfigurację zapisuje się w pliku */etc/pam.d/system-auth*, dodając dwie linie. Pierwsza z nich ma postać:

```
1 auth          required          pam_tally.so onerr=fail no_magic_root
```

Moduł *pam-tally* został wywołany z dwoma opcjami. Pierwsza z nich to **onerr**. Może mieć dwie wartości. Wartość *fail* powoduje, że do pliku dziennika zapisywane będą zdarzenia zakończone porażką (nieudane próby podłączenia do systemu). Wartość *success* spowoduje, że zapisywane będą zdarzenia zakończone powodzeniem. Opcja *no\_magic\_root* wymusza restrykcje również dla konta użytkownika *root*.

Druga linia to:

```
1 account      required      pam_tally.so deny=5 no_magic_root reset
```

W wywołaniu modułu pojawiły się dwie nowe opcje. Pierwsza z nich to **deny**, której wartość mówi o liczbie nieudanych prób autentykacji po których konto zostanie zablokowane. Opcja **reset** powoduje, że poprawne podłączenie do systemu usuwa listę historii połączeń niepoprawnych.

Informacja o nieudanych próbach podłączania się do systemu jest zapisywana do pliku *faillog*, który znajduje się w katalogu */var/log*. Pracę na zawartości pliku umożliwia komenda *faillog*. Do częściej wykorzystywanych opcji komendy należą:

- **-a** – wypisuje informacje dotyczące wszystkich użytkowników.
- **-m** *liczba\_prób* – pozwala na ustawienie liczby nieudanych prób podłączenia do systemu po których konto jest blokowane dla pojedynczego użytkownika (z opcją **-u**). Ustawienie wartości na 0, powoduje wyłączenie restrykcji.
- **-r** – zeruje licznik nieudanych prób podłączenia do systemu (najczęściej wykorzystywana z opcją **-u**).
- **-u** *nazwa\_użytkownika* – pozwala wskazać użytkownika, którego dotyczą operacje wykonywane przy pomocy komendy.

Ograniczmy liczbę nieudanych prób podłączenia do systemu dla użytkownika *antek* do 3.

```
1 [root@messy pam.d]# faillog -u antek -m 3
```

Historia nieudanych połączeń jest pusta:

```
1 [root@messy pam.d]# faillog -u antek
2 Login      Failures Maximum Latest      On
3
4 antek      0          3
```

Po 3 nieudanych próbach konto zostało zablokowane:

```
1 [root@messy pam.d]# faillog -u antek
2 Login      Failures Maximum Latest      On
3
4 antek      3          3  02/20/09 17:14:07 +0100  tty1
```

Konto może odblokować administrator systemu używając komendy:

```
1 [root@messy pam.d]# faillog -r -u antek
```

Po wykonaniu powyższej komendy, użytkownik *antek* może podłączyć się do systemu. Jego nieudane próby podłączenia do systemu są raportowane następująco:

```

1 [root@messy pam.d]# faillog -u antek
2 Login      Failures Maximum Latest      On
3
4 antek      0      3    02/20/09 17:14:07 +0100  tty1

```

### „Mocne” hasła

Praktycznie wszystkie programy służące do łamania haseł metodą brutalną działa w dwóch przebiegach. Pierwszy polega na szyfrowaniu słów ze słownika (najczęściej angielskiego) oraz haseł pochodzących z listy haseł „popularnych” i porównywaniu zgodności otrzymanej postaci zaszyfrowanej z przechowywaną w systemie. Drugi przebieg, znacznie bardziej czasochłonny, to generowanie ciągów znaków jako wszystkich kombinacji znaków z pewnego zdefiniowanego zbioru, a następnie ich szyfrowanie i porównywanie postaci zaszyfrowanej ze wzorcem. Czas odgadnięcia hasła znacznie kraca uwzględnienie konfliktów w algorytmach szyfrujących. Z kolei na wydłużenie czasu odgadnięcia możemy wpływać przedewszystkim wymuszając na użytkownikach, aby nie stosowali oni jako haseł słów pochodzących ze słownika. W tym celu wystarczy wymusić, aby hasło składało się nie tylko z liter, ale dodatkowo z cyfr i znaków specjalnych (\$, \_, , ...). Możemy również określić z ilu minimalnie znaków może być zbudowane hasło i w ten sposób wydłużymy czas trwania drugiego przebiegu programu.

Wprowadzenie restrykcji umożliwiają dwa alternatywne moduły: *pam\_cracklib* oraz *pam\_passwdqc*. W naszych przykładach zastosujemy moduł *pam\_cracklib*, chociaż ostatnio coraz częściej stosowanym jest moduł *pam\_passwdqc* głównie ze względu na nowe funkcjonalności. Definicja znajduje się w pliku *system\_auth*, znajdującym się w katalogu */etc/passwd*. Oryginalną linię:

```

1 password      requisite      pam_cracklib.so try_first_pass retry=3

```

modyfikujemy do postaci:

```

1 password      requisite      pam_cracklib.so try_first_pass retry=3
2 minlen=12 lcredit=1 ucredit=1 dcredit=1 ocredit=0

```

Znaczenie dodanych opcji jest następujące:

- **retry**=*n* – *n* oznacza liczbę prób zmiany hasła. Wartość domyślna to 1.
- **minlen**=*n* – ( $n \geq 0$ ) to minimalna długość hasła, określona jako suma liczby znaków tworzących hasło powiększona o liczbę punktów uzyskanych za użycie w hasle znaków z odpowiedniej grupy. Wartość domyślna to 9.
- **lcredit**=*n* – ( $n \geq 0$ ) oznacza maksymalną liczbę punktów przyznawaną za użycie w hasle małych liter. Jeśli w hasle użyto nie więcej niż *n* małych liter, każda daje jeden punkt do sumy. Wartość domyślna to 1. Jest ona zalecana wówczas, gdy wartość dla argumentu **minlen** jest nie większa niż 10.
- **ucredit**=*n* – ( $n \geq 0$ ) oznacza maksymalną liczbę punktów przyznawaną za użycie w hasle dużych liter. Jeśli w hasle użyto nie więcej niż *n* dużych liter, każda daje jeden punkt do sumy. Wartość domyślna to 1. Jest ona zalecana wówczas, gdy wartość dla argumentu **minlen** jest nie większa niż 10.

- **dcredit**= $n - (n \geq 0)$  oznacza maksymalną liczbę punktów przyznawaną za użycie w haśle cyfr. Jeśli w haśle użyto nie więcej niż  $n$  cyfr, każda daje jeden punkt do sumy. Wartość domyślna to 1. Jest ona zalecana wówczas, gdy wartość dla argumentu **minlen** jest nie większa niż 10.
- **ocredit**= $n - (n \geq 0)$  oznacza maksymalną liczbę punktów przyznawaną za użycie w haśle innych znaków. Jeśli w haśle użyto nie więcej niż  $n$  inny znak, każdy daje jeden punkt do sumy. Wartość domyślna to 1. Jest ona zalecana wówczas, gdy wartość dla argumentu **minlen** jest nie większa niż 10.
- **difok**= $n$  – to liczba znaków, którymi nowe hasło musi różnić się od starego, aby zostać zaakceptowanym. Wartość domyślna to 5. Jeśli natomiast co najmniej połowa znaków występująca w nowym haśle jest różna, to zostanie ono zaakceptowane bez względu na wartość argumentu.
- **dictpath**=ścieżka\_dostępu\_do\_pliku\_słownika – umożliwia podanie ścieżki dostępu do pliku z zabronionymi hasłami.

W naszym przykładzie minimalna liczba znaków w haśle wynosi 9. To 12 pomniejszone o 3 punkty, po jednym za użycie małych liter, dużych liter i cyfr. Użycie innych znaków nie jest premiowane punktami. Stąd w każdym haśle musi znaleźć się co najmniej jedna duża litera, jedna mała i jedna cyfra. W poniższym przykładzie hasła zostały dopisane.

```

1 [antek@messy ~]$ passwd
2 Changing password for user antek.
3 Changing password for antek.
4 (current) UNIX password: ala123
5 New UNIX password: AGU45p:
6 BAD PASSWORD: is too simple
7 New UNIX password: AGu45pwd2
8 Retype new UNIX password: AGu45pwd2
9 passwd: all authentication tokens updated successfully.
10 [antek@messy ~]$
```

## 2.10 Ograniczanie zasobów udostępnianych użytkownikom

Naturalnym dążeniem użytkowników jest maksymalne wykorzystanie dostępnych w systemie zasobów. W systemach wieloużytkownikowych należy zatem wykorzystać mechanizmy pozwalające na ich limitowanie w celu zapewnienia sprawiedliwego, adekwatnego do potrzeb udostępniania zasobów. Konkretnie wartości liczbowe są specyficzne dla każdego systemu i zależą od dostępnych zasobów, liczby korzystających z systemu użytkowników oraz charakteru uruchamianych zadań. Ustala się je na podstawie długotrwałych obserwacji. Pozwalają one określić średnie wykorzystanie zasobów i na jego podstawie określić limity. Nigdy nie przyjmuje się wartości wynikającej z podzielenia ilości dostępnego zasobu przez liczbę użytkowników. Rzadko zdarza się aby pracowali oni jednocześnie i maksymalnie wykorzystywali dany zasób. To właśnie wyniki obserwacji pozwalają skorygować tę wartość.

### 2.10.1 Mechanizmy dostępne w dystrybucji Fedora

W rodzinie systemów Red Hat ograniczenia dotyczące możliwości wykorzystania zasobów systemowych zapisane są w pliku *limits.conf* znajdującym się w katalogu */etc/security*. Prawo odczytu

zawartości ma każdy użytkownik w systemie, zaś modyfikacji jedynie użytkownik *root*. Plik ten jest plikiem tekstowym o budowie linijkowej. W jednej linii znajduje się jeden wpis limitujący. W pliku mogą pojawić się komentarze. Są nimi fragmenty linii od znaku *#* do końca linii.

Linia z wpisem składa się czterech pól oddzielonych co najmniej jednym znakiem białym. W kolejnych polach znajdują się:

1. Informacja o tym kogo dane ograniczenie dotyczy. W polu tym mogą znaleźć się:
  - nazwa użytkownika,
  - nazwa grupy użytkowników, poprzedzona znakiem *@*,
  - *\** oznaczająca wpis domniemany obowiązujący wszystkich,
2. Typ ograniczenia. Rozróżnia się dwa typy, a mianowicie oznaczane specyfikatorem **soft** ograniczenie, które może zostać przekroczone oraz ograniczenie typu **hard** niemożliwe do przekroczenia.
3. Oznaczenie limitowanego zasobu. W kolumnie tej mogą się pojawić następujące specyfikatory:
  - **core** – rozmiar pliku zawierającego obraz pamięci procesu tworzony m.in. wówczas, gdy proces sięgnął poza przydzielony mu obszar pamięci (tzw. *core*),
  - **data** – maksymalny rozmiar segmentu danych procesu w kB,
  - **fsize** – maksymalny rozmiar pliku w blokach danych w kB,
  - **memlock** – maksymalna liczba blokad adresów w pamięci,
  - **nofile** – maksymalna liczba otwartych plików,
  - **rss** – maksymalny rozmiar procesu w pamięci operacyjnej (ang. *Residual Set Size*) w kB,
  - **stack** – maksymalny rozmiar segmentu stosu w kB,
  - **cpu** – limit czasu procesora na jeden proces w minutach,
  - **nproc** – maksymalna liczba uruchomionych procesów,
  - **as** – ograniczenie rozmiaru przestrzeni adresowej,
  - **maxlogins** – maksymalna liczba jednoczesnych połączeń do systemu,
  - **priority** – wartość parametru *NICE* dla uruchamianych procesów,
  - **locks** – maksymalna liczba blokad na plikach.
4. Liczbową wartość ograniczenia.

Plik */etc/security/limits.conf* jest kopiowany podczas instalacji systemu. Zawiera on w formie komentarza opis składni oraz przykłady konfiguracji. Żadne ograniczenia zasobów nie są w nim aktywne. Wprowadzmy kilka przykładowych, przedstawionych poniżej:

```

1 @pracownicy hard priority 5
2 jan          hard core    4192
3 jan          soft cpu      2

```

W linii 1 zawarto ograniczenie dotyczące wartości parametru *NICE*. Proces każdego użytkownika będącego członkiem grupy *pracownicy* będzie miał wartość 5. Jego właściciel będzie mógł go zwiększać do maksymalnej wartości (teoretycznie 20) i zmniejszać do wartości 5. Druga linia ogranicza rozmiar pliku *core* dla użytkownika *jan* do 4192KB. Oba ograniczenia są typu *hard*. Trzecie ograniczenie dotyczy czasu procesora dla procesów użytkownika *jan*. Został on *miętko* ograniczony do 2 minut. Ograniczenia stają się aktywne dla sesji użytkownika otwartej po zapisaniu zmian w pliku */etc/security/limits.conf*. Informacje o nich można uzyskać komendą *ulimit* użytej z opcją *-a*:

```

1 [jan@messy ~]$ ulimit -a
2 core file size          (blocks, -c) 4192
3 data seg size          (kbytes, -d) unlimited
4 scheduling priority      (-e) 0
5 file size              (blocks, -f) unlimited
6 pending signals         (-i) 3052
7 max locked memory      (kbytes, -l) 32
8 max memory size        (kbytes, -m) unlimited
9 open files              (-n) 1024
10 pipe size              (512 bytes, -p) 8
11 POSIX message queues   (bytes, -q) 819200
12 real-time priority     (-r) 0
13 stack size             (kbytes, -s) 10240
14 cpu time               (seconds, -t) 120
15 max user processes     (-u) 1024
16 virtual memory         (kbytes, -v) unlimited
17 file locks             (-x) unlimited

```

Opis składa się z trzech kolumn. W pierwszej zawarty jest opis ograniczanego zasobu. W drugiej jednostka (opcjonalnie) oraz opcja komendy *ulimit* pozwalająca na ustalenie limitu dla danego zasobu. W kolumnie trzeciej znajduje się liczbową wartość ograniczenia lub słowo *unlimited* jeśli zasób nie jest limitowany. Ograniczenia zasobów komendą *ulimit* może dokonać każdy użytkownik. Oczywiście użytkownik *root* dokonuje tego w stosunku do każdego użytkownika w dowolny sposób zmieniając wartości ograniczeń. Zwykły użytkownik może wprowadzać jedynie ograniczenia bardziej restrykcyjne i to w stosunku tylko do swoich zasobów.

O wprowadzonych ograniczeniach parametru *NICE* przekonamy się listując procesy z bieżącej sesji. Wartość parametru określa zawartość kolumny opisanej jako *NI*:

```

1 [jan@messy ~]$ ps -l
2 F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
3 0 S   501  2220  2219  0  90   5 -  1200 wait  pts/1      00:00:00 bash
4 0 R   501  2336  2220  2  90   5 -  1137 -    pts/1      00:00:00 ps

```

Powstanie pliku *core* zostało wymuszone programem napisanym w języku C, w którym funkcją *free* zwalniano nie przydzieloną pamięć. W wyniku wykonania programu pojawiły się komunikaty:

```

1 [jan@messy ~]$ ./memory_test
2 *** glibc detected *** ./test: free(): invalid pointer: 0x00b0bdd0 ***
3 ===== Backtrace: =====
4 |/lib/libc.so.6[0xb8a7e4]

```

```

5 | /lib/libc.so.6(cfree+0x96) [0xb8c846]
6 | ./test[0x80483d0]
7 | /lib/libc.so.6(__libc_start_main+0xe6) [0xb335d6]
8 | ./test[0x8048321]
9 | ===== Memory map: =====
10 | 00110000-00111000 r-xp 00110000 00:00 0 [vdso]
11 | 00393000-003a0000 r-xp 00000000 08:02 214667 /lib/libgcc_s-4.3.0-20080428.so.1
12 | 003a0000-003a1000 rw-p 0000c000 08:02 214667 /lib/libgcc_s-4.3.0-20080428.so.1
13 | 00afd000-00b19000 r-xp 00000000 08:02 214657 /lib/ld-2.8.so
14 | 00b19000-00b1a000 r--p 0001c000 08:02 214657 /lib/ld-2.8.so
15 | 00b1a000-00b1b000 rw-p 0001d000 08:02 214657 /lib/ld-2.8.so
16 | 00b1d000-00c80000 r-xp 00000000 08:02 214658 /lib/libc-2.8.so
17 | 00c80000-00c82000 r--p 00163000 08:02 214658 /lib/libc-2.8.so
18 | 00c82000-00c83000 rw-p 00165000 08:02 214658 /lib/libc-2.8.so
19 | 00c83000-00c86000 rw-p 00c83000 00:00 0
20 | 08048000-08049000 r-xp 00000000 08:05 73741 /home/jan/test
21 | 08049000-0804a000 rw-p 00000000 08:05 73741 /home/jan/test
22 | 084dd000-084fe000 rw-p 084dd000 00:00 0 [heap]
23 | b80a1000-b80a3000 rw-p b80a1000 00:00 0
24 | bf9a8000-bf9bd000 rw-p bffeb000 00:00 0 [stack]
25 | Aborted (core dumped)

```

W katalogu bieżącym został utworzony plik z mapą pamięci:

```

1 | -rw----- 1 jan jan 294912 2009-04-28 12:51 core.2317

```

Najczęściej jest on wykorzystywany przez debugger do znajdowania błędów. Jeśli dopuszczalny rozmiar pliku *core* zostanie ustalony na 0 (jest to ustawienie domniemane), pliki te nie będą tworzone.

Uruchomienie procesu intensywnie wykorzystującego procesor, po przekroczeniu określonego limitu zakończyło się w następujący sposób:

```

1 | [jan@messy ~]$ ./prog
2 | CPU time limit exceeded (core dumped)

```

### 2.10.2 Mechanizmy dostępne w dystrybucji FreeBSD

W dystrybucji FreeBSD ograniczenia na przydział zasobów systemowych użytkownikom zapisywane są w pliku *login.conf* znajdującym się w katalogu */etc*. Należy zaznaczyć, że przy pomocy tego pliku możemy:

1. Precyzyjnie określać konfigurację dostępu do systemu.
2. Ograniczać wykorzystane zasobów systemu przez użytkowników.
3. Określać domyślne ustawienia środowiska.
4. Kontrolować opcje logowania i hasła.



Plik `/etc/login.conf` jest plikiem tekstowym. Jego zawartość może podglądać każdy użytkownik. Prawo modyfikacji posiada jedynie użytkownik `root`. System kontroli działa w oparciu o podział użytkowników na klasy logowania. Klasie logowania przypisywane są odpowiednie wartości atrybutów, które obowiązują wszystkich użytkowników do niej przypisanych. Plik `/etc/login.conf` rozpoczyna definicje dla klasy `default`, która jest wykorzystywana przez wszystkich użytkowników systemu bez jawnie przypisanej klasy logowania. Daje ona praktycznie nieograniczony dostęp do zasobów systemu. W pliku, w formie komentarza, pojawiły się przykładowe definicje klas logowania, które mogą stanowić podstawę do rozwiązań indywidualnych.

Plik ma budowę wierszową, tzn. wpis zawierający definicje atrybutów i ich wartości dla danej klasy logowania musi być zapisany w jednym wierszu. Dla poprawienia czytelności wiersz może zostać podzielony na linie. Jeśli na końcu linii pojawi się znak `\`, to oznacza on, że następna linia należy do bieżącego wiersza. Wiersz rozpoczyna nazwa klasy logowania zakończona `:"`. Następnie specyfikuje się kolejne atrybuty i ewentualnie nadane im wartości. Każda taka specyfikacja rozpoczyna się i kończy znakiem `:"`. Przykładowo fragment definicji atrybutów dla klasy logowania `default` ma postać:

```
1 default:\n2     :passwd_format=sha256:\n3     :priority=0:\n4     :requirehome:
```

Należy pamiętać, aby po dokonaniu modyfikacji w pliku `/etc/login.conf` dokonać aktualizacji bazy danych klas użytkowników. Służy do tego komenda `cap_mkdb`. Wymaga ona podania jako argumentu ścieżki dostępu do pliku `login.conf`. Spośród opcji komendy należy wymienić:

- **-b** – zapis do pliku bazy danych będzie zgodny z formatem *big-endian*.
- **-f** *ścieżka dostępu do pliku bazy danych* – domniemanym plikiem bazy danych kont użytkowników jest plik `login.conf.db` znajdujący się w katalogu `/etc`. Wartością opcji jest ścieżka dostępu do pliku alternatywnego.
- **-l** – zapis do pliku bazy danych będzie zgodny z formatem *low-endian*.
- **-v** – tryb gadatliwy polegający na wypisaniu liczby rekordów opisujących klasy logowania.

Znając budowę i zasady postępowania z plikiem `/etc/login.conf` przejdźmy do omówienia atrybutów.

### Ograniczenia zasobów

Atrybuty służące limitowaniu zasobów systemowych są następujące:

- **cputime** – maksymalna ilość czasu procesora na pojedynczy proces w minutach.
- **filesize** – maksymalny rozmiar pojedynczego pliku w kB.
- **datasize** – maksymalny rozmiar segmentu danych pojedynczego procesu w kB.
- **stacksize** – maksymalny rozmiar segmentu stosu pojedynczego procesu w kB.
- **coredumpsize** – maksymalny rozmiar pliku z obrazem pamięci procesu w kB.
- **memoryuse** – maksymalny rozmiar pamięci, jaki może zostać zajęty przez pojedynczy proces w kB.

- **maxproc** – maksymalna liczba procesów możliwa do uruchomienia przez użytkownika.
- **openfiles** – maksymalna liczba plików możliwych do otwarcia przez pojedynczy proces.
- **sbsize** – maksymalny rozmiar bufora gniazda sieciowego przydzielany pojedynczemu procesowi użytkownika.

Każdy atrybut może mieć wartość liczbową, lub jeśli opisywany zasób nie jest limitowany, symboliczną wartość *unlimited*. W pliku */etc/login.conf* można określić ograniczenie miękkie, nazywane w tej dystrybucji bieżącym oraz twarde określane jako maksymalne. Pierwsze z nich może zostać przekroczone i służy do poinformowania użytkownika o stopniu eksploatacji zasobów systemowych. Ograniczenie maksymalne nie może zostać przekroczone, a próba dokonania tego kończy się natychmiastowym zakończeniem procesu, który ograniczenie to próbował przekroczyć. Dla określenia ograniczenia bieżącego, do nazwy ograniczenia dodaje się przyrostek *-cur*. Przyrostek *-max* służy określaniu ograniczenia maksymalnego. Przykładowo:

```
1      :cputime-cur=75:\
2      :cputime-max=90:\
```

Jeśli jedno z ograniczeń nie zostanie podane, drugie przyjmuje wartość podanego.

### Określanie domyślnych ustawień środowiska

W pliku */etc/login.conf* można również definiować środowisko użytkownika należącego do danej klasy logowania. Niektórzy uważają, że mechanizm ten jest bardziej elastyczny od opisanego w rozdziale 5.3, a polegającego na zapisywaniu definicji w plikach konfiguracyjnych czytanych przez interpretery poleceń wszystkich użytkowników systemu lub pojedynczego użytkownika. Pozwala bowiem na konfigurowanie środowiska dla grupy użytkowników. Poniżej zamieszczono nazwy atrybutów oraz krótko opisano ich możliwości konfiguracyjne:

- **hushlogin** – występowanie tego atrybutu spowoduje, że podczas podłączania się użytkownika do systemu nie będą wypisywane informacje z pliku */etc/motd*.
- **ignorelogin** – umożliwia podłączenie się do systemu nawet wówczas, jeśli istnieje plik */var/run/nologin*.
- **manpath** – wartość to ścieżka dostępu do katalogu zawierającego pliki podręcznika systemowego (manuala).
- **nologin** – występowanie tego atrybutu powoduje zablokowanie możliwości podłączania się użytkowników należących do danej klasy logowania. Działanie jest analogiczne do wpisu w pliku */etc/login.access*.
- **path** – wartością atrybutu są ścieżki dostępu do katalogów, w których interpreter poleceń będzie poszukiwał programów do wykonania.
- **priority** – wartość atrybutu określa wartość minimalną parametru *NICE* z jakim będą uruchamiane procesy użytkowników.
- **requirehome** – występowanie atrybutu oznacza wymóg istnienia katalogu domowego dla każdego użytkownika z tej grupy logowania, który może podłączyć się do systemu.

- **setenv** – wartością atrybutu jest lista zmiennych środowiskowych. Zagadnienie zostało gruntownie omówione w rozdziale 5.3.
- **shell** – bezwzględna ścieżka dostępu do interpretera poleceń. Wartość zdefiniowana w pliku */etc/login.conf* przesłania wpis z pliku */etc/master.passwd*. Zdefiniowanie w obu plikach różnych interpreterów może prowadzić do anomalii w ich działaniu spowodowanych różną konfiguracją środowiska. Dotyczy to zwłaszcza interpreterów z rodziny *sh* oraz *csh*.
- **term** – wartość zmiennej to typ terminala. Wartość może być przesłonięta przez każdą aplikację ustawiającą typ terminala, bez żadnych konsekwencji w działaniu.
- **timezone** – domyślna wartość zmiennej środowiskowej *TZ* definiującej strefę czasową. Może zostać nadpisana.
- **umask** – wartość domyślna tzw. umaski wykorzystywanej przy określaniu domyślnych praw dostępu dla tworzonych plików i katalogów. Może zostać nadpisana.
- **welcome** – wartością atrybutu jest ścieżka dostępu do pliku zawierającego komunikat wypisywany podczas podłączania się użytkownika z danej klasy logowania do systemu. Wartością domyślną jest */etc/motd*.

### Kontrola opcji logowania i hasła

Z wykorzystaniem mechanizmów dostępnych w pliku */etc/login.conf* można również kontrolować różne opcje dotyczące haseł użytkowników oraz kontroli mechanizmu podłączania się użytkowników do systemu. Większość z tych ustawień jest dostępna tylko w tym pliku. Poniższa lista przedstawia te wykorzystywane najczęściej:

- **minpasswordlen** – wartość atrybutu określa minimalną liczbę znaków w hasle. Zmiana wartości atrybutu ujawni się przy najbliższej zmianie hasła. Wartości rozsądne wahają się od 7 do 10.
- **passwd\_format** – określa typ algorytmu stosowanego do kodowania lub szyfrowania haseł użytkowników należących do danej klasy logowania. W wersji 7 FreeBSD dostępne były trzy możliwości, a to: *des*, *md5*, *sha256*, *sha512* oraz *blf* oznaczająca *blowfish*. Pierwszy z nich jest domyślny i stosowany przy współpracy z innymi systemami operacyjnymi. Jego wadą, podobnie jak funkcji skrótu *md5* jest obecnie niska jakość szyfrowania objawiająca się relatywnie krótkim czasem odgadywania hasła. Jakościowo lepszy algorytm *blowfish* jest domniemanym w dystrybucji OpenBSD.
- **mixpasswordcase** – ustawienie tego atrybutu (nie posiada on wartości) spowoduje, że użytkownik nie będzie mógł ustawić hasła zbudowanego z jednego zestawu znaków, np. tylko z małych lub tylko z dużych liter.
- **copyright** – wartością atrybutu jest ścieżka dostępu do pliku z informacją o prawach autorskich do systemu.
- **host.allow** – jeśli atrybut zostanie tylko ustawiony, to będzie to oznaczało, że użytkownicy należący do danej klasy logowania będą mieli możliwość zdalnego podłączenia się do systemu tak, jak określono to w pliku */etc/hosts.allow*. Wartością atrybutu może być lista zawierająca adresy IP lub nazwy hostów oddzielone przecinkiem, z których użytkownicy będą mogli podłączać się zdalnie do systemu. Dla wyspecyfikowania całej sieci lub domeny

dopuszcza się stosowanie w adresie IP znaku "\*". Przykładowo *195.34.2.\** oznacza każdy host o adresie IP zaczynającym się na *195.34.2*

W takim ujęciu mechanizm służy do zawężania listy hostów, z których możliwe jest podłączenie do systemu, gdyż aby było ono możliwe nazwa lub adres IP hosta musi znajdować się na liście wartości atrybutu oraz w pliku */etc/hosts.allow*.

- **host.deny** – wartością atrybutu jest lista zawierająca nazwy lub adresy IP hostów oddzielone przecinkami, z których zdalne podłączenie się do systemu nie będzie możliwe. Dla wyspecyfikowania całej sieci lub domeny dopuszcza się w adresie IP stosowanie znaku "\*". Mechanizm ten w pewnym stopniu pokrywa się w pewnym zakresie z mechanizmem dostępnym poprzez plik */etc/login.access*.

Jeśli nazwa lub adres IP hosta występuje jednocześnie na liście wartości atrybutu **host.allow** oraz *host.deny*, to istotny jest wpis w **host.deny**.

- **times.allow** – atrybut pozwala zdefiniowanie okresów czasu, w których użytkownicy danej klasy logowania będą mogli podłączać się do systemu. Wartością jest lista dni tygodnia i przedziałów godzin oddzielonych przecinkami. Dni określane są przez dwuliterowy skrót nazwy angielskiej (kolejno od poniedziałku do niedzieli: *Mo, Tu, We, Th, Fr, Sa* oraz *Su*). Czas podawany jest w standardowej notacji 24-godzinnej, z dokładnością do godziny. Przykładowo wpis:

```
1 :times.allow=We13-24:\
```

oznacza możliwość podłączania się do systemu we środy w godzinach 13:00 do 24:00.

- **times.deny** – atrybut pozwala zdefiniowanie okresów czasu, w których użytkownicy danej klasy logowania nie będą mogli podłączać się do systemu. Wartości atrybutu specyfikuje się identycznie jak dla atrybutu **times.allow**. Należy pamiętać, że jeśli użytkownik jest już do systemu podłączony, to wejście w okres w którym nie ma możliwości podłączania się do systemu nie spowoduje jego odłączenia. Jeśli przedziały czasu określone jako wartości atrybutów **times.allow** oraz **times.deny** pokrywają się, to obowiązują ustawienia atrybutu **times.deny**.

### Ograniczanie dostępu do plików urządzeń

Plikiem, w którym można definiować sposób dostępu do urządzeń jest plik */etc/fstab*. Zawiera on informacje pozwalające na zmianę praw dostępu do plików urządzeń w momencie podłączania się użytkownika do systemu. Jest to plik tekstowy o budowie linjkowej. Jedna linia pozwala na zmianę praw dostępu do plików urządzeń dla przypadku podłączania się z konkretnego urządzenia. Dopuszcza się występowanie komentarza, który obowiązuje w linii od miejsca pojawienia się znaku # do znaku przejścia do nowej linii. Linie puste lub zawierające jedynie komentarz są ignorowane. Poniżej zamieszczono przykład pliku:

```
1 # FreeBSD: src/etc/fstab, v 1.3.56.1.4.1 2012/06/14 02:09:06 kensmith Exp $
2 #
3 /dev/ttyv0    0600 /dev/console
4 /dev/ttyv0    0000 /dev/*
```

Jak widać, każda czynna linia składa się z trzech kolumn oddzielonych od siebie co najmniej jednym znakiem białym. Pierwsza kolumna zawiera nazwę pliku reprezentującego urządzenie, z którego użytkownik podłącza się do systemu. Kolumna druga to prawa dostępu do plików urządzeń, zapisane w formacie ósemkowym. Trzecia kolumna to lista nazw plików urządzeń, którym zostaną nadane wyspecyfikowane prawa dostępu. Lista zawiera bezwzględne ścieżki dostępu do plików urządzeń, oddzielone znakiem `:`. W nazwach urządzeń dopuszcza się użycie znaku `*`, oznaczającego dowolny ciąg znaków z ciągiem pustym włącznie.

Reguła zapisana w linii 3 powyższego listingu mówi, że użytkownicy podłączający się z pierwszego terminala (numer 0) będą mieli prawo zapisu do i odczytu z urządzenia reprezentowanego plikiem `/dev/console`. W linii 4 zawarto zapis usuwający prawa dostępu do wszystkich urządzeń `/dev/lpt`, czyli praktycznie uniemożliwiający korzystanie z kolejek drukarkowych.

### Monitorowanie czynności użytkowników

**Uruchomianie i konfiguracja procesu audytu** Monitorowanie czynności użytkowników wymaga uruchomienia procesu działającego w tle (tzw. demona), który będzie śledził ich czynności i zapisywał informację do plików dziennika. Może to zrobić użytkownik `root` poleceniem:

```
1 [root@messy ~]# service auditd onestart
```

Rozwiązanie to ma tę wadę, że proces `auditd` nie będzie działał po ponownym uruchomieniu systemu. Jego uruchomienie podczas inicjalizacji systemu operacyjnego zapewni następujący wpis w pliku `/etc/rc.conf`:

```
1 auditd_enable=YES
```

Plikiem konfiguracyjnym definiującym sposób działania procesu `auditd` jest plik `/etc/security/audit_control`. Jest to plik tekstowy, w którym jedna linia definiuje wartość jednego argumentu. Składa się ona z dwóch kolumn oddzielonych znakiem `:`. Pierwsza kolumna zawiera nazwę parametru konfiguracyjnego, a druga listę wartości oddzielonych znakiem `,`. Dysponujemy następującymi parametrami konfiguracyjnymi:

- **dir** – wartość parametru to ścieżka dostępu do katalogu, w którym przechowywane są pliki audytu. Istnieje możliwość wyspecyfikowania kilku katalogów. Zmiana wartości tego parametru wymaga restartu procesu `auditd`.
- **flags** – parametr pozwala wyspecyfikować nazwy klas zdarzeń, które będą śledzone dla wszystkich użytkowników. W pliku `/etc/security/audit_user` istnieje możliwość wyspecyfikowania klas zdarzeń dla inspekcji konkretnego użytkownika.
- **host** – wartością parametru jest nazwa hosta lub jego adres IP w wersji IV lub VI. Użycie nazwy spowoduje, że uruchamiany proces inspekcji `auditd` zechce na jej podstawie uzyskać adres IP. Stąd konieczność odpowiedniego wpisu w pliku `/etc/hosts` lub zdefiniowania adresu IP serwera DNS, który dokona tłumaczenia nazwy na adres IP (podstawowy konfiguracji sieciowych zostały omówione w tomie drugim). Jeśli uzyskanie adresu IP hosta nie będzie możliwe, proces audytu nie uruchomi się. Nazwa hosta pojawi się w nagłówkach rekordów zapisywanych do pliku audytu oraz w nazwach plików audytu.
- **naflags** – wartością parametru jest lista flag audytu pozwalających zdefiniować klasy zdarzeń podlegające audytowi, jeśli dana akcja nie została przypisana do konkretnego użytkownika.

- **minfree** – parametr pozwala określić minimalny dostępny obszar systemu plików konieczny do zapisywania plików audytu. Zmniejszenie dostępnego rozmiaru poniżej określonego powoduje zapisanie do plików audytu ostrzeżenia. Wartość domyślna, jeśli nie zostanie określona parametrem, wynosi 20% dostępnego rozmiaru systemu plików.
- **policy** – wartością parametru jest lista flag polityki globalnego audytu. Ogólnie, zawartość listy decyduje o poziomie audytu systemu. Domyślnie używa się flag *cnt* oraz *argv*. Dostępne flagi opisano poniżej.
- **filesz** – wartością opcji jest maksymalny rozmiar pliku audytu, po przekroczeniu którego program audytu dokona jego rotacji. Rotacja polega na przesunięciu istniejących plików archiwalnych o jeden poziom „głębiej” oraz zapisaniu bieżącego pliku ze zdarzeniami pochodzącymi z audytu jako pierwszego archiwalnego. Rozmiar mniejszy niż 512 kB jest traktowany jako błędny i w takiej sytuacji przyjmuje się wartość 0, która oznacza, iż automatyczna rotacja plików nie będzie przeprowadzana. Podanie rozmiaru pliku jako liczby niemianowanej oznacza, iż rozmiar w bajtach. Dodatkowo przyjęto następujące oznaczenia jednostek: *B* – bajty, *K* – kilobajty, *M* – megabajty oraz *G* – gigabajty. Stąd oznaczenia 8388608 oraz 8M są jednoznaczne.
- **expire-after** – wartość opcji określa moment usunięcia pliku audytu. Wartością opcji może być okres czasu, który minął od ostatniej operacji zapisu zdarzenia do pliku, sumaryczny rozmiar istniejących plików audytu lub kombinacja obu tych wartości.

Wartością opcji *policy* mogą być następujące flagi definiujące polityki audytu:

- **cnt** – flaga opisuje zachowanie procesu w przypadku wyczerpania się miejsca przeznaczonego do zapisywania informacji o audycie. Ustawienie pozwala na działanie programu w sytuacji, gdy zdarzenia nie są zapisywane. Jeśli flaga nie została ustawiona, to wykonanie procesu zostaje zawieszone.
- **ahlt** – ustawienie flagi spowoduje zatrzymanie systemu operacyjnego w przypadku, gdy nie będzie możliwe przeprowadzenie audytu danego zdarzenia. Zatrzymanie zostanie poprzedzone zapisaniem do pliku audytu rekordu opisującego dane zdarzenie.
- **argv** – ustawienie flagi pozwala śledzić argumenty wywołania funkcji systemowej *execve*.
- **arge** – flaga pozwala na śledzenie zmiennych środowiskowych przekazanych jako argumenty do funkcji *execve*.
- **seq** – flaga umożliwia załączenie unikalnego identyfikatora sekwencji do rekordu opisującego dane zdarzenie. W implementacji w systemie FreeBSD flaga ta jest ignorowana.
- **group** – flaga odpowiada za załączenie listy grup dodatkowych użytkownika do rekordu opisującego dane zdarzenie. W systemie FreeBSD brak jest implementacji możliwości obsługi również tej flagi. Lista grup dodatkowych nie jest nigdy załączana.
- **trail** – flaga pozwala na dołączenie do każdego rekordu zapisywanego do pliku audytu stopki zawierającej numer identyfikacyjny zdarzenia. W systemie FreeBSD obsługa tej flagi nie została zaimplementowana. Stopka dodawana jest zawsze.
- **patch** – flaga pozwala na dołączanie do rekordów audytu ścieżek dostępu do plików pomocniczych, przechowujących dodatkowych informacje. Opcja ta nie została zaimplementowana w systemie FreeBSD. Ścieżki dostępu do plików pomocniczych nie są dołączane do rekordów audytu.

- **zonename** – użycie flagi wymusza dołączanie do rekordu audytu identyfikatora strefy. W systemie FreeBSD nie została zaimplementowana obsługa tej flagi. Identyfikator strefy nie jest dołączany do rekordów audytu.
- **perzone** – ustawienie flagi pozwala na prowadzenie audytu każdej strefy osobno. Implementacja systemu audytu w systemie FreeBSD nie umożliwia wykorzystania tej flagi. Stąd wszystkie rekordy audytu są kierowane do jednego strumienia.

**Zarządzanie procesem audytu** Do zarządzania systemem audytu służy polecenie *audit*. Jego uruchomienie wymaga użycia opcji, która określi sposób jego działania. Dostępne są następujące opcje:

- **-e** – opcja wymusza natychmiastowe usunięcie plików audytu spełniających warunki, które pozwalają na ich usunięcie (rozmiar, zakres czasowy audytu) zgodnie z konfiguracją zapisaną w pliku */etc/security/audit\_control*.
- **-i** – opcja ta nie została zaimplementowana w systemie FreeBSD. W innych systemach korzystających z tej wersji systemu audytu służy do jego inicjalizacji i uruchamiania.
- **-n** – użycie tej opcji wymusza na systemie audytu zamknięcie bieżącego pliku audytu oraz dokonanie rotacji istniejących plików, zgodnie z konfiguracją zawartą w pliku */etc/security/audit\_control*. Pliki audytu, które spełniają warunki ich usunięcia zostaną usunięte.
- **-s** – opcję tę wykorzystuje się wówczas, gdy zmianie uległa konfiguracja systemu audytu. Jej użycie powoduje synchronizację opcji systemu audytu do aktualnej konfiguracji zapisanej w pliku */etc/security/audit\_control*. Dodatkowo zostaje utworzony nowy plik audytu.
- **-t** – opcja wymusza zakończenie pracy systemu audytu. Pliki audytu zostają zamknięte, a ich nazwy zmienione tak, aby ułatwić odczyt z nich czasu zamknięcia systemu.

Użycie programu *audit* będzie możliwe, jeśli uruchomiony będzie proces *auditd*. Wymagane jest również, aby użytkownik używający go należał do grupy użytkowników *audit*.

**Monitorowanie użytkowników** Podstawowym plikiem konfiguracyjnym, definiującym monitorowanie poszczególnych użytkowników jest plik */etc/security/audit\_user*. To plik tekstowy o budowie linijkowej. W pliku mogą pojawić się komentarze, które zaczynają się znakiem *#*, a kończą znakiem przejścia do nowej linii. Linie puste są ignorowane. Linie konfiguracyjne składają się z trzech pól oddzielonych znakiem *:* i dotyczą pojedynczego użytkownika. Przykładowy plik ma postać:

```
1 #
2 root:lo:no
3 jan:lo,fr,+pc,ot:no
```

Pierwsze pole w linii zawiera nazwę użytkownika. Drugie specyfikatory klas zdarzeń, oddzielone przecinkami, które będą zawsze zapisywane w zienniku. Trzecie to specyfikatory klas zdarzeń nigdy nie podlegających audytowi. Specyfikator klasy zdarzenia może zostać poprzedzony znakiem *+*, który oznacza, że zapisywaniu będą jedynie zdarzenia z tej klasy zakończone sukcesem. Znak *-* spowoduje, że inspekcji podlegać będą zdarzenia zakończone niepowodzeniem. Pojawienie się samego specyfikatora skutkuje zapisaniem zdarzenia z danej klasy bez względu na sposób jego zakończenia.

W powyższym przykładzie, w linii 2 zdefiniowano zapisywanie informacji o podłączeniach do systemu i odłączeniach użytkownika *root*, niezależnie od tego, czy zakończyły się one sukcesem czy niepowodzeniem. Żadne inne zdarzenia nie podlegają audytowi. W przypadku użytkownika *jan* audytowi podlegają: podłączanie się do systemu i odłączanie, odczyty plików zakończone sukcesem i niepowodzeniem, próby uruchamiania procesów zakończone sukcesem oraz pozostałe, niezdefiniowane w innych klasach zdarzenia. Wszystkie, pozostałe klasy zdarzeń nie będą podlegały audytowi.

Opis audytowanych klas zdarzeń zawiera plik */etc/security/audit\_class*. Jest to plik tekstowy o budowie linijkowej. Znakiem komentarza, jak zwykle jest znak *#*. Za komentarz uważa się znaki w linii od znaku *#* do przejścia do nowej linii. Poniżej zamieszczono przykładowy plik *audit\_class*:

```

1 # $P4: //depot/projects/trustedbsd/openbsm/etc/audit_class#6 $
2 # $FreeBSD: release/9.1.0/contrib/openbsm/etc/audit_class 191273 2009-04-19 16:17:13Z rw
3 atson $
4 #
5 0x00000000:no:invalid class
6 0x00000001:fr:file read
7 0x00000002:fw:file write
8 0x00000004:fa:file attribute access
9 0x00000008:fm:file attribute modify
10 0x00000010:fc:file create
11 0x00000020:fd:file delete
12 0x00000040:cl:file close
13 0x00000080:pc:process
14 0x00000100:nt:network
15 0x00000200:ip:ipc
16 0x00000400:na:non attributable
17 0x00000800:ad:administrative
18 0x00001000:lo:login_logout
19 0x00002000:aa:authentication and authorization
20 0x00004000:ap:application
21 0x20000000:io:ioctl
22 0x40000000:ex:exec
23 0x80000000:ot:miscellaneous
24 0xffffffff:all:all flags set

```

Jak widać każde monitorowane zdarzenie należy do właściwej klasy zdarzeń. Każda linia definicyjna składa się z trzech kolumn oddzielonych znakiem *:*. Odzworowuje ona maskę monitorowanego zdarzenia (pierwsza kolumna), na jego klasę (kolumna druga) oraz opis.

Na zakończenie najistotniejsza kwestia, czyli przeglądania plików zawierających informacje z systemu monitorowania. Są to pliki binarne przechowywane w katalogu, którego nazwa stanowi wartość parametru *dir* zdefiniowanego w pliku */etc/security/audit\_control*. W katalogu tym przechowywane są pliki zgodnie ze zdefiniowaną polityką rotacji. Na bieżący plik pokazuje dowiązanie symboliczne o nazwie *current*. Nazwę pliku bieżącego stanowi data jego utworzenia w formacie *rokmięsiećdzieńgodzinaminutasekunda.not\_terminated*. Pliki historyczne posiadają nazwy, które składają się z daty ich utworzenia, znaku *.* oraz daty ich rotacji. Dodatkowo, w nazwie każdego z plików może pojawić się po znaku *.* nazwa hosta. Poniżej zamieszczono przykładową zawartość katalogu z plikami audytu:



```

1 -r--r----- 1 root  audit    56 Sep 17 16:35 20130917143456.20130917143520
2 -r--r----- 1 root  audit   2602 Sep 25 15:04 20130917143520.20130925130413
3 -r--r----- 1 root  audit    113 Sep 25 15:04 20130925130414.20130925130430.messy
4 -r--r----- 1 root  audit    929 Sep 25 15:09 20130925130509.20130925130915.messy
5 -r--r----- 1 root  audit    929 Sep 25 15:15 20130925130921.20130925131550.messy
6 -r--r----- 1 root  audit   1793 Sep 28 14:54 20130925131557.20130928125441
7 -r--r----- 1 root  audit    336 Sep 28 15:00 20130928125449.20130928130013
8 -r--r----- 1 root  audit  12037 Oct  6 03:01 20130928130018.not_terminated
9 lrwxr-xr-x  1 root  audit    40 Sep 28 15:00 current -> /var/audit/20130928130018.not_terminated

```

Do przeglądania zawartości plików audytu służy polecenie *praudit*. Nazwę pliku audytu można podać jako argument polecenia lub skierować jego zawartość na standardowe wejście. Domyślnie, zawartość pliku audytów postaci tekstowej jest kierowana na standardowe wyjście. Polecenie posiada kilka użytecznych opsi, do których należą:

- **-d separator** – opcja umożliwia zdefiniowanie separatora dla wypisywanych wartości. Domyślnie jest to przecinek.
- **-l** – umożliwia wypisanie zawartości jednego rekordu w jednej linii.
- **-p** – opcję tę używa się w przypadku, gdy zawartość pliku audytu jest kierowana na standardowe wejście z wykorzystaniem polecenia *tail*. Umożliwia ona wypisanie informacji pochodzącej od następnego (kompletnego) rekordu.
- **-r** – opcja umożliwia wypisanie pełnej informacji z rekordów pliku audytu w postaci numerycznej. Wyklucza się z opcją **-s**.
- **-s** – opcja umożliwia wypisanie informacji z rekordów pliku audytu w postaci symbolicznej. Przykładowo, numery UID zostaną zastąpione nazwami logowania, pojawią się opisy zdarzeń zapisane w pliku */etc/security/audit\_event*. Wyklucza się z opcją **-r**.
- **-x** – opcja umożliwia wypisanie informacji z pliku audytu w formacie xml.

Poniżej przedstawiono wybrane zdarzenia z pliku audytu w postaci tekstowej:

```

1 header,87,11,login - local,0,Mon Oct 14 15:49:30 2013, + 928 msec
2 subject,-1,root,wheel,-1,-1,1404,4294967295,0,0.0.0.0
3 text,Login incorrect
4 return,failure : Operation not permitted,2
5 trailer,87
6 header,110,11,open(2) - read,0,Mon Oct 14 15:49:34 2013, + 880 msec
7 argument,2,0x0,flags
8 path,/home/jan/.login_conf.db
9 subject,jan,jan,jan,root,wheel,1404,1404,0,0.0.0.0
10 return,failure : No such file or directory,4294967295
11 trailer,110
12 header,133,11,open(2) - read,0,Mon Oct 14 15:49:34 2013, + 880 msec
13 argument,2,0x0,flags
14 path,/etc/login.conf.db
15 attribute,644,root,wheel,99,963147,1928944

```

```
16 | subject,jan,jan,jan,root,wheel,1404,1404,0,0.0.0.0  
17 | return,success,6  
18 | trailer,133
```

Pierwsze zdarzenie, o identyfikatorze 87 opisuje niepoprawne podłączanie się do systemu użytkownika *root*. Kolejne, o identyfikatorze 110 to próba odczytania przez użytkownika *jan* nieistniejącego pliku. Na koniec zdarzenie opisujące poprawne otwarcie pliku przez użytkownika *jan*. Należy zwrócić uwagę, iż przytoczone zdarzenia są wybranymi. Faktycznie otwarcie pliku przez użytkownika poprzedzone jest kilkoma innymi zdarzeniami, jak np. analizą możliwości wejścia do kolejnych katalogów znajdujących się w ścieżce dostępu. Stąd liczba zapisanych rekordów audytu związanego z odczytem pliku wynosi co najmniej kilka.

## Rozdział 3

# Dyskowe systemy plików

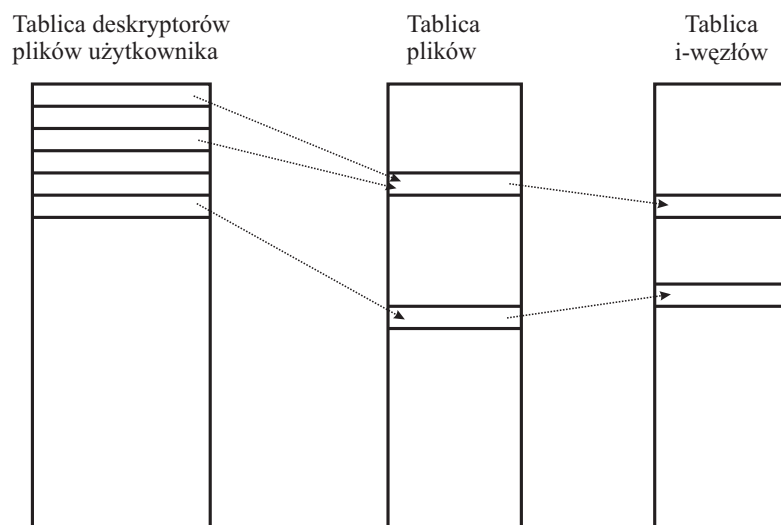
### 3.1 Przegląd podsystemu plików

Reprezentację pliku w systemie operacyjnym udostępnia struktura nazywana *i-węzłem*. Zawiera ona między innymi identyfikatory właścicieli pliku (indywidualnego i grupowego), prawa dostępu, czasy dostępu oraz opis położenia pliku na dysku. Każdy plik jest opisany przez jeden *i-węzeł*, lecz może mieć kilka nazw. Wszystkie one odnoszą się do jednego *i-węzła*. Każdą nazwę określa się jako dowiązanie. Proces odwołuje się do pliku za pomocą nazwy będącej ścieżką dostępu określającą jego położenie w drzewie katalogów. Jądro analizuje kolejne jej składniki będące nazwami katalogów i sprawdza, czy proces ma prawo do przechodzenia do katalogów znajdujących się w ścieżce dostępu. W końcu uzyskuje *i-węzeł* pliku i z wykorzystaniem zapisanej w niej informacji wykonuje operacje wymagane przez proces. *I-węzły* przechowywane są w systemie plików, lecz podczas obsługi plików jądro czyta je do tablicy w pamięci operacyjnej.

Do pracy z systemami plików jądro wykorzystuje jeszcze dwie tablice. Są to: *tablica deskryptorów plików użytkownika* oraz *tablica plików*. Tablica deskryptorów plików użytkownika jest przydzielana każdemu procesowi. Tablica plików jest globalną strukturą jądra. Kiedy proces otwiera lub tworzy plik, jądro przydziela w każdej tablicy miejsce odpowiadające *i-węzłowi* pliku. Informacje pamiętane w tych trzech tablicach opisują stan pliku i sposób dostępu do niego. W tablicy plików pamiętane jest przesunięcie względem początku pliku do bajtu, od którego rozpocznie się wykonywanie następnej operacji zapisu lub odczytu. Pamiętane są również prawa dostępu procesu, który plik otworzył. W tablicy deskryptorów przechowywane są numery wszystkich plików otwartych przez proces.

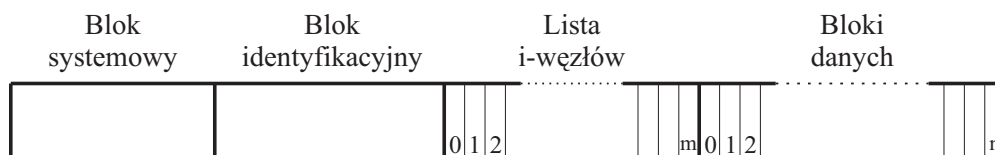
Sposób odwoływania się procesu do pliku przedstawiono schematycznie na rysunku 3.1. Z wykorzystaniem deskryptora (numeru) pliku odczytywany jest rekord z tablicy deskryptorów plików. Z rekordu odczytywany jest numer pozycji opisującej plik w tablicy plików. Stąd pochodzi informacja o stanie operacji zapisu lub odczytu wykonywanych na pliku, jak również o numerze opisującego go *i-węzła*. Na podstawie danych przechowywanych w *i-węzle* jądro wykonuje żądane przez proces operacje, jeśli zdefiniowane prawa dostępu to umożliwiają.

System operacyjny Unix przechowuje pliki zwykle oraz katalogi na urządzeniach blokowych. W zdecydowanej większości przypadków są to dyski. Na dysku może znajdować się kilka systemów plików, jak również system plików może zostać utworzony na kilku dyskach. Jądro pracuje na poziomie logicznym, realizując współpracę raczej z systemami plików, a nie z dyskami i traktuje każdy z nich jako urządzenie logiczne identyfikowane numerem. Za konwersję adresu urządzenia logicznego, czyli systemu plików na adres urządzenia fizycznego, odpowiada podprogram obsługi dysku zwany popularnie sterownikiem.



Rysunek 3.1: Struktury danych jądra systemu operacyjnego wykorzystywane do pracy na plikach.

W literaturze można spotkać kilka definicji systemów plików. Najbardziej ogólne określają system plików jako format przechowywania informacji. W takim ujęciu system plików zawiera *metadane* oraz *dane*. Metadane stanowią opis struktur i formatu przechowywania danych w systemie. Mówią nam w jaki sposób zorganizowana jest wewnętrzna struktura systemu plików. Dane to zawartość plików. Inna grupa definicji mówi, że system plików składa się z ciągu bloków logicznych, zawierających 512, 1024, 2048 lub dowolną inną wielokrotność 512 bajtów, zależnie od implementacji systemu. Wydaje się, że definicja ta powstała jako próba opisu budowy logicznej pierwotnego systemu plików systemu Unix, nazywanego czasem *s5*. Schemat jego budowy przedstawiono na rysunku 3.2.



Rysunek 3.2: Budowa logiczna pierwotnego uniksowego systemu plików *s5*.

System plików *s5* posiada następującą strukturę:

- **Blok systemowy** znajduje się na początku systemu plików. Zajmuje on zazwyczaj pierwszy sektor i może zawierać program ładujący, służący do inicjalizacji systemu operacyjnego. Chociaż do załadowania systemu operacyjnego wystarcza tylko jeden blok systemowy, to w tym rozwiązaniu każdy system plików ma taki blok (może on być pusty).
- **Blok identyfikacyjny** (inaczej *superblok*) opisuje system plików. Informacja przechowywana w superbloku to rozmiar systemu plików, liczba możliwych do utworzenia w nim plików, lista wolnych i-węzłów, lista wolnych bloków danych itd. Przy montowaniu systemu plików jądro wczytuje jego superblok do pamięci, a zapisuje go na dysku po każdym

wywołaniu funkcji systemowej *sync* oraz przy odmontowaniu systemu plików.

- **Lista i-węzłów** zawiera i-węzły umieszczone w systemie plików za blokiem identyfikacyjnym. Rozmiar listy określany jest podczas tworzenia systemu plików. Jądro odwołuje się do i-węzłów za pomocą indeksu w liście i-węzłów. Jeden i-węzeł jest i-węzłem głównym systemu plików i udostępnia strukturę katalogów systemu plików po wykonaniu montowania (funkcja systemowa *mount*).
- **Bloki danych** znajdują się za listą i-węzłów i zawierają dane z plików oraz niektóre metadane. W klasycznej implementacji blok danych może należeć do tylko jednego pliku w systemie plików.

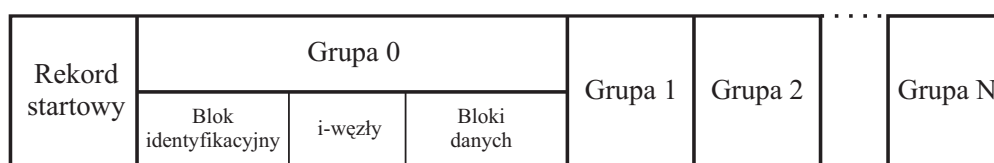
Zaproponowany system plików był rozwiązaniem bardzo zaawansowanym, jak na czasy w których powstał. Umożliwiał stosowanie 14-to znakowych nazw plików oraz obsługiwał pliki i partycje o dużych rozmiarach. Jego słabą stroną była wydajność. Wynika ona przede wszystkim z małego rozmiaru bloku danych. Oryginalna implementacja używała bloków o rozmiarze 512 B. Później został on powiększony do 1024 i 2048 B. To jednak nie wystarczyło do poprawienia wydajności. Zauważmy, że żądanie odczytania bloku danych z systemu plików kierowane do dysku wymaga w pierwszej kolejności ustawienia głowic nad ścieżką, na której znajduje się dany blok. Następnie należy poczekać, aż blok znajdzie się pod głowicą. Następuje odczyt danych do momentu przeczytania ostatniego sektora zawierającego dane bloku. Statystycznie czas oczekiwania na pojawienie się bloku pod głowicą, to czas wykonania przez dysk połowy obrotu. Przy prędkości 15.000 obrotów na minutę daje to 2 ms. Stąd w ciągu jednej sekundy można przeczytać 500 bloków. Przy rozmiarze bloku wynoszącym 1 kB uzyskujemy teoretyczną prędkość 0.5 MB/s. Przy transferze wewnętrznym dochodzącym do 150 MB/s stanowi to ułamek procenta wydajności dysku. Przedstawione szacunki są pesymistyczne. Faktycznie system może odczytać więcej niż jeden blok ze ścieżki, gdy tylko dojdzie do właściwego sektora. Będzie to możliwe wówczas, gdy bloki zawierające kolejne fragmenty danych zostały zapisane w kolejnych blokach na dysku. System s5 stara się tak rozmieszczać dane, ale w miarę przydzielania i zwalniania bloków zawartość listy wolnych bloków danych staje się całkowicie losowa. Zatem, po pewnym czasie używania systemu plików, kolejne bloki przydzielane jednemu plikowi rzadko będą leżały obok siebie. Powoduje to opóźnienie przy dostępie związane z szybkością obrotową dysku oraz dodatkowo z ruchem głowic między ścieżkami. Stąd dążenie do ograniczania fragmentacji plików. Szybkim, ale nieeleganckim sposobem podniesienia wydajności jest zwiększenie rozmiaru bloków danych. Wiąże się to jednak z marnotrawieniem miejsca w systemie plików. Wynika ono z faktu, że liczba bajtów zapisana w pliku rzadko jest wielokrotnością rozmiaru bloku danych. Przy założeniu, że w bloku danych mogą być przechowywane dane należące do jednego pliku, blok przechowujący ostatni fragment danych zazwyczaj nie jest wykorzystany w całości.

Inny poważny problem dotyczący wydajności wynika z lokalizacji i-węzłów na dysku. Jak widać znajdują się one na początku systemu plików i zajmują kilka procent rozmiaru. W praktyce oznacza to, że przy każdym dostępie do pliku, który nie był dawno otwierany, czyli jego i-węzeł nie znajduje się w pamięci podręcznej, głowice dysku wędrują na jego początek. Po odczytaniu zawartości i-węzła przesuwają się do wskazanego w nim bloku. W najszybszych dyskach głowica przesuwa się w czasie 1–4 ms, co jak widać jest operacją długotrwałą.

Należy także zwrócić uwagę na fakt, że system plików s5 jest bardzo wrażliwy na awarię. Jest to wynikiem zastosowania mechanizmu opóźnionego zapisu realizowanego przez pamięć buforów dyskowych. Stąd jeśli system ulegnie awarii zanim zawartość buforów zostanie przeniesiona na dysk, to traci się zawartość zmodyfikowanych plików oraz metadanych. Problem jest tym istotniejszy, że bufor pamięci zapisywane są w przypadkowej kolejności. Może zatem zaistnieć sytuacja polegająca na tym, że bloki danych przechowujące informacje danego pliku zostaną

uaktualnione, natomiast opisujący plik i-węzeł już nie. W systemie plików mogą zatem pojawić się „osierocone” bloki danych i i-węzły, które należy zwolnić, gdyż nie można ustalić do jakich plików one należą. Trafiają one do katalogu o nazwie *lost+found*. Rozwiązania tego problemu są dwa. Pierwsze, mniej eleganckie, polega na zrezygnowaniu z mechanizmu buforów dyskowych. Drugie, bardziej zaawansowane, to wprowadzenie mechanizmu dziennikowania, pozwalające na przynajmniej częściowe odzyskiwanie danych oraz uspoźnianie zawartości systemu plików po awarii.

Przed wszystkim wymienione problemy z wydajnością stały się podstawą do opracowania przez twórców systemu BSD alternatywnego systemu plików, który został włączony do wersji BSD4.1. System został nazwany *Fast File System* (FFS) dla zwrócenia uwagi na dbałość autorów o jego wydajność. Jego robocza nazwa to *ufs*. Źródła sukcesu należy upatrywać w sprytnym rozmieszczeniu i-węzłów oraz bloków danych na dysku. Budowę logiczną systemu plików FFS przedstawiono na rysunku 3.3.



Rysunek 3.3: Budowa logiczna systemu plików FFS.

Jak widać, w systemie plików FFS zastosowano podział na tzw. grupy cylindrów. Zostały one wprowadzone w celu minimalizacji czasu przesuwania głowic przy odczycie danych z i-węzła, a następnie danych opisującego go pliku. FFS stara się bowiem przechowywać i-węzeł, bloki danych oraz katalog pliku w tej samej grupie cylindrów. Utrzymując mały rozmiar grupy cylindrów w stosunku do rozmiaru partycji, na której założono system plików, dodatkowo ogranicza się konieczność przesuwania głowic. Z drugiej strony, aby algorytmy wyszukiwania wolnych bloków danych działały skutecznie, w obrębie grupy cylindrów musi istnieć odpowiedni obszar wolnego miejsca. W przeciwnym razie system byłby zmuszony wyboru nieoptymalnego, polegającego na przydzieleniu bloków danych dla pliku w innej grupie cylindrów. W tym przypadku optymalizacja polega na tym, że ostatnie 10% wolnego miejsca w systemie plików nie jest dostępne dla zwykłych użytkowników.

Kolejnym rozwiązaniem wpływającym na podniesienie wydajności jest użycie większego rozmiaru bloku danych. W pierwotnym rozwiązaniu było to 4 kB, a obecne implementacje wykorzystują bloki o rozmiarach 8-64 kB. Aby rozwiązać problem marnowania miejsca przy dużym rozmiarze bloku danych, w systemie FFS używa się pojęcia fragmentu, którego rozmiar może wahać się od 1/8 do rozmiaru całego bloku danych. Fragment jest najmniejszą jednostką alokacji, zatem wykorzystanie miejsca w blokach danych nie jest gorsze niż w systemie s5. Rozmiar fragmentu ustala się w momencie tworzenia systemu plików, a w istniejących systemach plików nie można go niestety zmieniać.

Dla zwiększenia niezawodności FFS zapisuje wszystkie metadane w sposób synchroniczny bez buforowania. Oznacza to, że w praktyce po awarii systemu prawdopodobieństwo utraty i-węzłów, nazw plików, czy katalogów jest znacznie mniejsze. Dodatkowo na dysku znajduje się kilka kopii superbloku, zatem nawet w przypadku utraty kopii z grupy 0, programami narzędziowymi można przywrócić dowolną kopię z jednej z pozostałych grup cylindrów. Stąd utrata pojedynczego superbloku nie spowoduje utraty zawartości całego systemu plików, jak miało to miejsce w systemie s5.

## 3.2 Informacja przechowywana w i-węźle

Opis budowy oraz zasad działania systemu plików rozpoczniemy od poznania podstawowych atrybutów plików. Wartości atrybutów dostarcza nam komenda *ls* użyta z opcjami: **-a** – wypisuje informacje o wszystkich plikach z ukrytymi włącznie, **-l** – informacja jest dostarczana w postaci długiej oraz **-i** – na początku linii opisującej plik pojawia się numer definiującego go i-węzła (odpowiada on pozycji na liście i-węzłów). Poniżej przedstawiono fragment zawartości katalogu uzyskany w ten właśnie sposób:

```

1 wacek@thorin $ ls -ali
2 total 176
3 210913 drwx----- 4 wacek wacek 4096 Jun  8 11:46 .
4      2 drwxr-xr-x  7 root  root  4096 Mar  4  2008 ..
5 210924 -rw----- 1 wacek wacek 3492 Jun  8 11:46 .bash_history
6 210917 -rw-r--r-- 1 wacek wacek   24 Mar  4  2008 .bash_logout
7 227142 -rw-r--r-- 1 wacek wacek  227 Apr 18 13:32 .bash_profile
8 227143 -rw-r--r-- 1 wacek wacek  146 Apr 18 13:34 .bashrc
9 210932 drwxrwxr-x 2 wacek wacek 4096 Apr 18 13:12 binaria
10 1233025 -rw-rw-r-- 1 wacek wacek    0 May 26 13:26 derqewru
11 210919 drwxr-xr-x 3 wacek wacek 4096 Mar  4  2008 .kde
12 210927 -rwxrwxr-x 1 wacek wacek 4664 Mar 28 11:11 prog
13 210931 -rw-rw-r-- 1 wacek wacek  108 Mar 28 11:11 prog.c
14 227138 -rwxrwxr-x 1 wacek wacek 5055 Mar 28 11:12 signal
15 210934 -rw-rw-r-- 1 wacek wacek  345 Nov 19  2004 signal.c
16 227141 -rw----- 1 wacek wacek 1877 Mar 28 13:46 .viminfo
17 210914 -rw-r--r-- 1 wacek wacek  658 Mar  4  2008 .zshrc

```

Listing rozpoczyna linia z informacją ile bloków dyskowych zajmują dane należące do plików z bieżącego katalogu (opcja **-l**). Kolejne linie opisują pliki i w przypadku użycia opcji **-il** składają się z ośmiu pól. Pola oznaczają:

1. Numer i-węzła opisującego dany plik.
2. Rodzaj pliku oraz prawa dostępu do niego.
3. Liczbę dowiązań twardych.
4. Nazwę użytkownika w systemie będącego właścicielem indywidualnym pliku.
5. Nazwę grupy w systemie, która jest właścicielem grupowym.
6. Rozmiar pliku podawany w bajtach (użycie opcji **-h** (od human) powoduje wypisanie rozmiaru w formacie łatwiejszym do odczytania, zwłaszcza przy dużych rozmiarach. Jest on podawany w kB, MB, GB, TB, EB).
7. Datę ostatniej modyfikacji pliku.
8. Nazwę pliku.

W kolejnych punktach omówiono ich znaczenie oraz sposoby i możliwości modyfikacji.

### 3.2.1 Typ pliku

Typ pliku opisuje pierwszy od strony lewej znak w drugiej kolumnie przytoczonego listingu. Jak wiemy, istotną zaletą systemu Unix jest to, iż używa on spójnego formatu plików traktując każdy plik jako strumień bajtów. Możemy zatem powiedzieć, że każdy obiekt występujący w systemie plików to plik. Na potrzeby systematyki rozróżniamy trzy podstawowe typy plików. Są to:

1. Pliki zwykłe.
2. Katalogi.
3. Pliki specjalne.

Plik zwykły to taki, w którym system operacyjny nie widzi żadnej struktury wewnętrznej. Jego zawartość nie jest przez system operacyjny w żaden sposób interpretowana. Ten typ pliku oznaczany jest znakiem `-`.

Katalog posiada dla systemu operacyjnego strukturę wewnętrzną. Zawiera on informacje o plikach znajdujących się w nim nadając tym samym systemowi plików strukturę hierarchiczną. Dane przechowywane w katalogu mają postać pozycji składających się z numeru i-węzła i nazwy pliku zawartego w katalogu. Katalog jest oznaczany literą `d`.

Pliki specjalne to kilka typów plików. Do tej grupy należą:

- Pliki reprezentujące urządzenia znakowe. Pliki te znajdują się w katalogu `/dev` i stanowią reprezentacje tzw. urządzeń surowych. Oznaczone są one literą `c`.
- Pliki reprezentujące urządzenia blokowe. Pliki reprezentują urządzenia, z którymi komunikacja odbywa się blokami danych z wykorzystaniem mechanizmu buforowania. Ten typ pliku oznaczany jest literą `b`.

Pliki reprezentujące urządzenia można utworzyć przy pomocy komendy `mknod`. Jej składnia minimalnie różni się pomiędzy dystrybucjami. W rodzinie RedHat wymagane jest podanie typu pliku (`b` lub `c`), nazwy pliku oraz liczby tzw. dużej (*major*) i małej (*minor*) określających odpowiednio numer sekcji kodu w jądrze systemu operacyjnego oraz fragmentu w obrębie sekcji odpowiedzialnego za obsługę danego urządzenia. Tworzonemu plikowi można nadać prawa dostępu przy pomocy opcji `-m`. Jeśli opcja nie zostanie użyta prawa zostaną nadane w oparciu o wartość umaski. Właścicielem indywidualnym pliku będzie użytkownik, który wywołał komendę, zaś grupowym jego grupa podstawowa. Komendę może wykonać z sukcesem jedynie użytkownik `root`. Przykładowo komenda:

```
1 [root@thorin ~]# mknod dysk b 10 5
```

stworzy w katalogu bieżącym plik o następujących wartościach atrybutów:

```
1 [root@thorin ~]# ls -l dysk
2 brw-r--r-- 1 root root 10, 5 Jun  8 15:04 dysk
```

Składnia oraz sposób działania komendy `mknod` w dystrybucji FreeBSD są nieco inne. Komenda wymaga podania nazwy tworzonego pliku, typu pliku (`b` lub `c`) oraz liczb *major* i *minor*. Komendę może wykonać z sukcesem jedynie użytkownik `root`. Oczywiście z wykorzystaniem opcji `-m` można określić prawa dostępu do tworzonego pliku. Dodatkowo można określić właściciela indywidualnego i grupowego. Przykładowo:



```
1 # mknod cd0 c 0 0 wacek:bin
```

stworzy w katalogu bieżącym plik o następujących wartościach atrybutów:

```
1 # ls -l cd0
2 crw-r--r-- 1 wacek bin 0, 0 Jun 8 17:49 cd0
```

- Plik reprezentujący urządzenie znakowe niebuforowane. Został on wprowadzony w dystrybucji RedHat i jest oznaczany literą *u*. Plik tego typu można utworzyć również ręcznie poleceniem *mknod*.
- Plik typu *pipe*, oznaczany literą *p*. Jest wykorzystywany do komunikacji między dwoma procesami, z których jeden zapisuje komunikat do pliku, a drugi odczytuje. Plik tego typu można utworzyć „ręcznie” za pomocą polecenia *mkfifo*. Funkcja ta w różnych dystrybucjach posiada praktycznie taką samą składnię. Istotną opcją jest **-m** umożliwiające nadanie praw dostępu do tworzonego pliku. Jeśli nie zostanie ona użyta, to prawa zostaną nadane zgodnie z obowiązującą umaską.

```
1 wacek@thorin $ mkfifo pipe
2 wacek@thorin $ ls -l pipe
3 prw-rw-r-- 1 wacek wacek 0 Jun 8 14:35 pipe
```

- Plik typu *socket*, podobnie jak *pipe*, jest wykorzystywany w komunikacji międzyprocesowej. Oznaczony jest literą *s*. Plik tego typu tworzy się funkcją systemową *socket*, zaś wysyłanie i odbieranie komunikatu od innego procesu realizuje się odpowiednio funkcjami *sendmsg* oraz *recvmsg*.
- Plik będący dowiązaniem symbolicznym oznaczany jest literą *l*. Plik taki zawiera ścieżkę dostępu do pliku oryginalnego. Tworzy się go komendą *ln*. Opis i porównanie dowiązań symbolicznych i twardego stanowią treść kolejnego podpunktu.

Przedstawione powyżej typy plików występują praktycznie w każdej dystrybucji systemu Unix. Mogą oczywiście pojawić się typy specyficzne, jak chociażby wspomniany typ *u* w dystrybucji RedHat, czy wprowadzony jedynie w systemie SUN Solaris typ oznaczony literą *D* (door), służący komunikacji międzyprocesowej w modelu klient-serwer. Jak widać zakres ich zastosowania jest bardzo wąski. Stąd zainteresowanych należy raczej odesłać do dokumentacji konkretnego systemu.

### Dowiązania twarde i symboliczne

W większości systemów plików zaimplementowane zostały dwa rodzaje dowiązań, nazywane symbolicznymi i twardymi. Dla zbadania ich właściwości posłużymy się następującym przykładem. W katalogu bieżącym znajdują się katalogi o nazwach *katalog1* oraz *katalog2*, a w nich odpowiednio pliki *uwagi.txt* oraz *ls.bin*. Sytuację ilustruje wynik wywołania komendy *ls* z opcjami **-ilR**:

```
1 $ ls -ilR
2 total 4
3 47117 drwxr-xr-x 2 wacek 1001 512 Jun 9 13:31 katalog1
```

```

4 47118 drwxr-xr-x  2 wacek  1001  512 Jun  9 13:31 katalog2
5
6 ./katalog1:
7 total 18
8 47119 -rw-r--r--  1 wacek  1001 18042 Jun  9 13:31 uwagi.txt
9
10 ./katalog2:
11 total 26
12 47120 -r-xr-xr-x  1 wacek  1001 25228 Jun  9 13:31 ls.bin

```

Służące do tworzenia dowiązań polecenie *ln* domyślnie tworzy dowiązanie twarde. Stworzenie dowiązania symbolicznego wymaga użycia opcji *-s*. Komenda wymaga również podania jednego argumentu, którym jest nazwa (ścieżka dostępu) pliku, do którego dowiązanie będzie stworzone. Wykonanie tak uruchomionej komendy spowoduje utworzenie w katalogu bieżącym pliku, będącego dowiązaniem o nazwie identycznej z nazwą pliku, do którego dowiązanie było tworzone. Jeśli dowiązanie ma mieć inną nazwę, to musimy użyć drugiego argumentu, którym jest właśnie ta nazwa.

Stworzymy po dwa dowiązania do każdego z plików. Dla pliku *uwagi.txt* będą to dowiązania symboliczne o nazwach *symbol1* oraz *symbol2* utworzone w katalogu *katalog2*. W przypadku pliku *ls.bin* będą to dowiązania twarde o nazwach *twardy1* oraz *twardy2* utworzone w katalogu *katalog1*. Dowiązania utworzono następującymi poleceniami:

```

1 $ cd katalog2
2 $ ln -s ../katalog1/uwagi.txt symbol1
3 $ ln -s ../katalog1/uwagi.txt symbol2
4 $ cd ../katalog1
5 $ ln ../katalog2/ls.bin twardy1
6 $ ln ../katalog2/ls.bin twardy2

```

Zwróćmy uwagę na pierwsze różnice między dowiązaniami, które uwidaczniają się już na etapie tworzenia dowiązań:

1. Stworzenie dowiązania symbolicznego jest możliwe nawet wówczas, jeśli plik do którego tworzymy dowiązanie nie istnieje. Powstały plik będący dowiązaniem pokazuje wówczas na nieistniejący plik. Można go „podrzucić” w dowolnym momencie. Dla dowiązania twardego plik do którego jest on tworzony musi istnieć.
2. Dowiązanie symboliczne możemy tworzyć do dowolnego typu pliku. Dowiązania twardego nie możemy tworzyć do katalogów.
3. Dowiązanie symboliczne może być tworzone do pliku znajdującego się w dowolnym miejscu drzewa katalogów. Dowiązanie twarde może być stworzone do pliku, który znajduje się w tym samym systemie plików. Uwaga stanie się jasna po lekturze podrozdziału dotyczącego administracji systemami plików.

Po utworzeniu dowiązań sprawdzamy zawartość katalogów testowych:

```

1 $ ls -lR
2 total 4
3 47117 drwxr-xr-x  2 wacek  1001  512 Jun  9 13:44 katalog1

```

```

4 47118 drwxr-xr-x  2 wacek  1001  512 Jun  9 13:44 katalog2
5
6 ./katalog1:
7 total 70
8 47120 -r-xr-xr-x  3 wacek  1001 25228 Jun  9 13:31 twardy1
9 47120 -r-xr-xr-x  3 wacek  1001 25228 Jun  9 13:31 twardy2
10 47119 -rw-r--r--  1 wacek  1001 18042 Jun  9 13:31 uwagi.txt
11
12 ./katalog2:
13 total 26
14 47120 -r-xr-xr-x  3 wacek  1001 25228 Jun  9 13:31 ls.bin
15 47121 lrwxr-xr-x  1 wacek  1001      21 Jun  9 13:44 symbol1 -> ../katalog1/uwagi.txt
16 47122 lrwxr-xr-x  1 wacek  1001      21 Jun  9 13:44 symbol2 -> ../katalog1/uwagi.txt

```

Wnioski z obserwacji są następujące:

1. Plik *uwagi.txt* z katalogu *katalog1* oraz pliki *symbol1* i *symbol2* z katalogu *katalog2*, będące jego dowiązaniami symbolicznymi to różne pliki. Różne numery i-węzłów świadczą o tym, że opisują one różne pliki. Dodatkowo pliki te są widoczne pod jedną nazwą każdy, o czym mówi liczba z trzeciej kolumny linii listingu opisująca te pliki. Rozmiar pliku *uwagi.txt*, do którego utworzono dowiązania jest różny od rozmiaru plików będących dowiązaniami. Ich rozmiary w naszym przypadku wynoszą 21 bajtów i jest to liczba znaków w ścieżce dostępu do pliku będącego oryginałem. Ona właśnie stanowi zawartość dowiązań symbolicznych.
2. Plik *ls.bin* z katalogu *katalog2* oraz pliki *twardy1* i *twardy2* z katalogu *katalog1* to te same pliki. Wszystkie one są opisane przez ten sam i-węzeł. Każdy z nich jest widoczny pod trzema nazwami. Rozmiar każdego z nich jest identyczny.

Przyjrzyjmy się wykorzystaniu bloków danych oraz i-węzłów w systemie plików, w którym utworzono dowiązania. Informacje można uzyskać komendą *df*:

```

1 $ df -i /home
2 Filesystem 1K-blocks Used Avail Capacity iused ifree %iused Mounted on
3 /dev/ad0s1e 1982798 128 1824048 0% 31 259039 0% /home

```

Jak widać w systemie plików zajętych jest 128 bloków danych oraz 31 i-węzłów. Usuwamy plik *uwagi.txt* z katalogu *katalog1*, posiadający dwa dowiązania symboliczne. Wykorzystanie zasobów systemu plików przedstawia się wówczas następująco:

```

1 Filesystem 1K-blocks Used Avail Capacity iused ifree %iused Mounted on
2 /dev/ad0s1e 1982798 110 1824066 0% 30 259040 0% /home

```

Usunięto jeden plik, więc liczba zajętych i-węzłów zmniejszyła się o jeden. Rozmiar pliku wynosił 18042 bajty, czyli zajmował on 18 bloków o rozmiarze 1024 bajty. Stąd zmniejszenie liczby zajmowanych bloków o 18. Niestety, w przypadku dowiązania symbolicznego dane znajdują się jedynie w pliku, do którego utworzono dowiązania. Po jego usunięciu są tracone.

W dalszej kolejności usuniemy plik *ls.bin* z katalogu *katalog2*, posiadający dwa dowiązania twarde. Stan systemu plików jest teraz następujący:

```

1 $ df -i /home
2 Filesystem 1K-blocks Used Avail Capacity iused ifree %iused Mounted on
3 /dev/ad0s1e 1982798 110 1824066 0% 30 259040 0% /home

```

Jak widać liczba zajętych bloków danych oraz i-węzłów nie uległa zmianie, a dane z pliku *ls.bin* są nadal dostępne w plikach będących dowiązaniem twardym. Dopiero usunięcie ich spowoduje zwolnienie zasobów do następującego poziomu:

```

1 $ df -i /home
2 Filesystem 1K-blocks Used Avail Capacity iused ifree %iused Mounted on
3 /dev/ad0s1e 1982798 84 1824092 0% 29 259041 0% /home

```

Zwolnionych zostało 26 bloków danych. Rozmiar pliku wynosił 25288 bajtów, czyli zajmował on 25 bloków po 1024 bajty. Stąd jeden blok danych został wykorzystany do przechowania informacji o nazwach pliku. Został zwolniony jeden i-węzeł. Na tej podstawie można wysnuć wniosek, iż dowiązanie twarde nie tworzy fizycznej kopii bloków danych, a jedynie pamięta nazwy pod którymi występuje plik w systemie plików. Usunięcie pliku będącego dowiązaniem twardym sprowadza się do usunięcia jego nazwy. Dopiero usunięcie ostatniej nazwy powoduje zwolnienie bloków danych oraz i-węzła opisującego plik.

Reasumując, dowiązania symboliczne oraz twarde zajmują praktycznie tyle samo miejsca w systemie plików. Dowiązanie twarde posiada ograniczenia dotyczące dowiązywania katalogów oraz plików znajdujących się w różnych systemach plików. Jest natomiast znacznie bardziej bezpieczne, gdyż dane z pliku są dostępne do momentu usunięcia jego ostatniej nazwy.

Kilka słów na temat opcji komendy *ln*. Otóż w dystrybucjach RedHat oraz FreeBSD posiadają niemal identyczne opcje. Do najczęściej wykorzystywanych w dystrybucji RedHat należą:

- **-f** – jeśli jako drugi argument komendy, a więc nazwę pliku będącym dowiązaniem podano nazwę pliku, który istnieje, to komenda przed stworzeniem dowiązania usuwa jego zawartość.
- **-F** – usiłuje stworzyć dowiązanie twarde do katalogu. Opcja ta jest dostępna jedynie dla użytkownika *root* oraz w nielicznych typach systemach plików. Praktycznie nie jest wykorzystywana.
- **-i** – praca w trybie interaktywnym. Przed usunięciem istniejącego pliku, którego nazwa została podana jako nazwa dowiązania, komenda wymaga zatwierdzenia tej operacji.
- **-s** – tworzy dowiązanie symboliczne. Domyślnie tworzone jest dowiązanie twarde.
- **-v** – przełącza komendę w tryb gadatliwy.

Najczęściej wykorzystywanymi opcjami komendy *ln* w dystrybucji FreeBSD są:

- **-f** – usuwa zawartość istniejącego pliku, jeśli jego nazwa została podana jako nazwa dowiązania (drugi argument wywołania komendy).
- **-F** – jeśli jako nazwa dowiązania została podana nazwa istniejącego katalogu, opcja usuwa go i tworzy plik o identycznej nazwie będący dowiązaniem. Najczęściej opcje tę wykorzystuje się z opcjami **-f** oraz **-i**.

- **-h** – opcja ta jest wykorzystywana jeśli tworzone jest dowiązanie, którego nazwa jest identyczna z nazwą istniejącego dowiązania symbolicznego do katalogu. Użycie tej opcji sprawi, że nie zostanie utworzone dowiązanie we wskazywanym katalogu, a jedynie podmieniony plik, na który istniejące dowiązanie pokazuje. Opcja **-n** działa analogicznie, a została wprowadzona dla zachowania zgodności z innymi wersjami.
- **-i** – opcja wykorzystywana z opcjami **-f** oraz **-F**. Powoduje interaktywne działanie komendy w przypadku konieczności usunięcia pliku.
- **-s** – użycie tej opcji spowoduje, że w wyniku działania komendy zostanie utworzone dowiązanie symboliczne. Domyślnie tworzone jest dowiązanie twarde.
- **-v** – opcja powoduje raportowanie każdego tworzonego dowiązania.

Przykładem zastosowania dowiązania twardego są komendy *chpass*, *chfn* oraz *chsh* z systemu FreeBSD. Faktycznie to jeden plik występujący pod tymi nazwami, co przedstawiono na poniższym listingu. Usunięcie dowolnej z nich powoduje, że dana komenda nie jest dostępna, ale pozostałe nadal działają.

```

1 $ ls -li 'which chpass'
2 800893 -r-sr-xr-x 6 root wheel 18468 Feb 24 2008 /usr/bin/chpass
3 $ ls -li 'which chfn'
4 800893 -r-sr-xr-x 6 root wheel 18468 Feb 24 2008 /usr/bin/chfn
5 $ ls -li 'which chsh'
6 800893 -r-sr-xr-x 6 root wheel 18468 Feb 24 2008 /usr/bin/chsh

```

### 3.2.2 Prawa dostępu

Kolejne dziewięć znaków z opisu pliku określa prawa dostępu do pliku. Zaczniemy od teorii. Otóż w dużym skrócie sprawy bezpieczeństwa systemów komputerowych, w tym prawa dostępu, reguluje dokument *Trusted Computer System Evaluation Criteria* (TCSEC). Jest to standard Departamentu Obrony Stanów Zjednoczonych. Znany pod nazwą *Pomarańczowa książka*, został po raz pierwszy opublikowany w roku 1983 przez *National Computer Security Center* (NCSC), a następnie poprawiony w roku 1985. Obecnie zastąpił go dokument *Common Criteria for Information Technology Security Evaluation*, który obowiązuje od roku 2005. Zawarta w nim klasyfikacja kontroli dostępu wyróżnia następujące jej typy:

- *Mandatory Access Control* (MAC) – to sposób kontroli dostępu, w którym system operacyjny wymusza na aktywnym zasobie systemu komputerowego (podmiocie) możliwość dostępu lub wykonania pewnych operacji na danym przedmiocie. Aktywnym zasobem jest najczęściej proces lub wątek. Podmiotami mogą być plik, katalog, port TCP, segment pamięci współdzielonej, itp. Zarówno podmiot jak i przedmiot posiadają zbiór atrybutów bezpieczeństwa. Są one wykorzystywane przez reguły działające w jądrze systemu operacyjnego, które określają możliwość wykonania danych operacji przez podmiot na przedmiocie. Czynność ta jest wykonywana przed rozpoczęciem operacji. Polityki bezpieczeństwa w systemie są kontrolowane centralnie przez administratora bezpieczeństwa. Użytkownicy nie mają możliwości ich modyfikacji. Rozwiązanie to zostało zaimplementowane w projektach SELinux<sup>1</sup> oraz TrustedBSD<sup>2</sup>.

<sup>1</sup>Opis projektu można znaleźć na stronie <http://www.nsa.gov/research/selinux/>. Indywidualne implementacje dla każdej dystrybucji są dostępne na ich stronach www, np. [www.redhat.com](http://www.redhat.com)

<sup>2</sup>Strona www projektu dostępna pod adresem <http://www.trustedbsd.org>

- *Discretionary Access Control* (DAC) – ten sposób kontroli dostępu przekazuje uprawnienia do zarządzania danym obiektem w ręce jego właściciela. Najczęściej jest nim użytkownik lub grupa użytkowników. W przypadku próby wykonania operacji na obiekcie, rola systemu operacyjnego sprowadza się do sprawdzenia, czy proces lub wątek wykonujący ją oraz obiekt na którym jest wykonywana mają wspólnego właściciela oraz czy posiada on uprawnienia do wykonania tej operacji. Rozwiązanie to jest powszechnie wykorzystywane w systemach uniksowych zarówno w klasycznej implementacji, jak również w coraz powszechniej stosowanych listach kontroli dostępu (*Access Control Lists*, ACL).

Alternatywą do metod kontroli dostępu MAC oraz DAC jest *Role-Based Access Control* (RBAC). Polega ona na umożliwianiu dostępu jedynie autoryzowanym użytkownikom. Praktyczna realizacja polega na przypisaniu możliwości wykonania danej operacji zdefiniowanym w systemie rodom. Użytkowników systemu przypisuje się do określonych ról dając im w ten sposób możliwość wykonywania tych operacji. Sposób ten jest na tyle elastyczny, że umożliwia pracę zarówno jako MAC jak i DAC. Został zaimplementowany w projektach SELinux oraz TrustedBSD. Dokładny opis można znaleźć w kolejnym tomie, w rozdziałach poświęconych tym właśnie projektom oraz w licznych publikacjach, np. w [4, 7].

### Tradycyjne prawa dostępu

Podstawowy mechanizm kontroli dostępu w większości systemów operacyjnych z rodziny Unix bazuje na DAC. W praktyce oznacza to, że każdy obiekt ma właściciela, który decyduje o prawach dostępu. Omawianie zaczniemy od tradycyjnej implementacji w zastosowaniu do systemu plików, a następnie przejdziemy do bardziej elastycznego rozwiązania, a mianowicie list kontroli dostępu.

W podejściu tradycyjnym każdy obiekt w systemie plików posiada właściciela indywidualnego oraz grupowego. Właścicielem indywidualnym jest najczęściej użytkownik, który ten obiekt stworzył, zaś grupowym jego grupa podstawowa. Ponieważ to właściciel ma prawo decydowania o prawach dostępu do obiektu, operacja zmiany właściciela jest poddana dość mocnym restrykcjom. Prawa dostępu do obiektu zostały podzielone na trzy grupy, które dotyczą: właściciela indywidualnego, właściciela grupowego oraz wszystkich pozostałych użytkowników w systemie. Prawa dla grup są rozłączne. Zatem prawa dla właściciela grupowego dotyczą wszystkich użytkowników będących członkami tej grupy poza właścicielem indywidualnym, jeśli on do grupy tej również należy. Prawa dla pozostałych użytkowników systemu dotyczą wszystkich użytkowników systemu poza użytkownikiem będącym właścicielem indywidualnym oraz użytkownikami należącymi do grupy będącej właścicielem grupowym.

W obrębie każdej grupy praw wprowadzono trzy podstawowe prawa dostępu. Prawa te zapisywane są na tej samej pozycji. W kolejności od lewej są to:

1. Prawo odczytu oznaczane literą **r**, jeśli jest ustawione lub znakiem – jeśli zostało cofnięte.
2. Prawo zapisu oznaczane literą **w**, jeśli jest ustawione lub znakiem – jeśli zostało cofnięte.
3. Prawo wykonania oznaczane literą **x**, jeśli jest ustawione lub znakiem – jeśli zostało cofnięte.

Stąd dziewięć znaków występujących w opisie pliku po oznaczeniu jego typu to trzy prawa dostępu dla właścicieli indywidualnego, grupowego i pozostałych użytkowników systemu. Przykładowo prawa dostępu **rwxr-xr---** oznaczają, że użytkownik będący właścicielem indywidualnym ma prawo odczytu, zapisu i wykonania, użytkownicy należący do grupy będącej właścicielem grupowym mają prawo odczytu i wykonania, wszyscy pozostali użytkownicy mają jedynie prawo odczytu.

Znaczenie poszczególnych praw jest zależne od typu pliku. W tym przypadku różnicujemy je na katalogi i pozostałe typy plików. W przypadku plików nie będących katalogami znaczenie praw jest następujące:

- Prawo odczytu, jeśli jest ustawione, umożliwia czytanie zawartości pliku. W praktyce oznacza to możliwość pomyślnego wykonania komend np. *cat* czy *less* z argumentem będącym nazwą pliku, czy wczytanie jego zawartości do edytora.
- Prawo zapisu, jeśli jest ustawione, umożliwia modyfikowanie zawartości pliku.
- Prawo wykonywania, jeśli jest ustawione, umożliwia potraktowanie danego pliku jak programu, który może zostać wykonany – interpreter poleceń podejmie próbę zlecenia jego wykonania do jądra sterującego operacyjnego.

W przypadku katalogów znaczenie praw dostępu należy interpretować następująco:

- Prawo odczytu daje możliwość czytania zawartości katalogu. W praktyce wykonanie komendy *ls* dla tego katalogu zakończy się sukcesem.
- Prawo zapisu pozwala modyfikować zawartość katalogu. Skoro katalog to plik zawierający informacje o znajdujących się w nim plikach, to prawo zapisu daje praktycznie możliwość tworzenia nowych i usuwania istniejących plików. Jeśli natomiast plik już istnieje, to jego zawartość może być modyfikowana bez posiadania prawa zapisu do katalogu, w którym się on znajduje. Chcąc zabezpieczyć plik przed usunięciem należy cofnąć prawo zapisu do katalogu, w którym plik ten się znajduje.
- Prawo wykonania dla katalogu oznacza, że może on stać się katalogiem bieżącym lub inaczej możemy do tego katalogu przejść. Wynikiem struktury katalogu jest powiązanie praw odczytu oraz wykonania. Otóż listowanie zawartości katalogu, do którego mamy prawo odczytu, a nie mamy prawa wykonania wygląda w systemie Fedora następująco:

```

1 [antek@thorin ~]$ ls -ali katalog
2 total 0
3 ??----- ? ? ? ?          ? .
4 ??----- ? ? ? ?          ? ..
5 ??----- ? ? ? ?          ? opis_projekty.txt

```

System FreeBSD w takiej sytuacji nie próbuje nawet realizować wypisywania informacji w postaci długiej, ograniczając się jedynie również do informacji o nazwach zawartych w nim plików:

```

1 $ ls -ali katalog
2 ls: .: Permission denied
3 ls: ..: Permission denied
4 ls: opis_projekty.txt: Permission denied
5 total 0

```

Przedstawione powyżej, tradycyjne prawa dostępu stanowią mechanizm bardzo sztywny, który w pewnych wypadkach może prowadzić do obniżenia poziomu bezpieczeństwa systemu. Przykładem może być polityka odnośnie haseł użytkowników. Z jednej strony administrator powinien wymuszać odpowiednią częstotliwość ich zmian. Z drugiej hasła są zapisywane w plikach,





Działanie prawa w odniesieniu do katalogów jest nieco inne. Jeśli katalog ma ustawione prawo SGID, to właścicielem grupowym każdego tworzonego w nim pliku jest grupa będąca właścicielem grupowym tego katalogu. Mówiąc bardziej obrazowo, właściciel tworzonego pliku jest dziedziczony po katalogu. Prawo to jest wykorzystywane w implementacji współdzielenia plików przez użytkowników należących do jednej grupy. Tworzony jest wówczas katalog, którego właścicielem grupowym zostaje grupa użytkowników pracująca na wspólnych plikach. Katalogowi ustawia się prawo SGID. Każdy utworzony w nim plik, jako właściciela grupowego będzie miał tę grupę użytkowników. Zapewni to użytkownikom do niej należącym dostęp zgodnie z prawami obowiązującymi właściciela grupowego.

- Prawo **SVTX** (*sticky bit*, bit lepki) zapisane jest w trójce praw dostępu obowiązujących pozostałych użytkowników systemu. Obowiązuje dla katalogów. Praktycznie oznacza, że plik z danego katalogu może usunąć jedynie jego właściciel (oraz oczywiście użytkownik *root*). Jest ono stosowane dla katalogów wspólnych, w których wszyscy użytkownicy mogą przechowywać swoje pliki. Jeśli tak, to katalog taki musi mieć ustawione prawo zapisu dla pozostałych użytkowników. Ustawienie tego prawa umożliwia jednocześnie usuwanie plików pozostałych użytkowników. Przed tym zabezpiecza prawo SVTX. Klasyczny przykładem jest katalog */tmp*, gdzie tymczasowo wszyscy użytkownicy mogą przechowywać swoje pliki bez obawy o usunięcie je przez innego użytkownika.

Nadawanie praw dostępu dla nowo tworzonych plików odbywa się przy uwzględnieniu tzw. *umaski*, którą kontrolujemy poleceniem *umask*. Jest to polecenie wbudowane interpreterów poleceń *csh*, zaś zewnętrzne dla interpreterów poleceń z rodziny *sh*. Ustalanie wartości umaski odbywa się najczęściej w plikach konfiguracyjnych interpretera poleceń, co zostało opisane w rozdziale 5.3. Można oczywiście modyfikować ją z linii komend. Wywołanie komendy bez argumentu powoduje wypisanie bieżącej wartości umaski. Wartość ta mówi, których praw dostępu nie nadajemy dla tworzonego pliku. Przykładowo:

```
1 -bash-4.0$ umask
2 0022
```

Umaska jest zapisana w postaci czterech cyfr. Skrajna lewa dotyczy rozszerzonych praw własności, kolejna praw dostępu dla właściciela indywidualnego, następnie dla właściciela grupowego i skrajna prawa dla pozostałych użytkowników systemu. W zapisie tym prawa mają swoje wartości. I tak prawo SUID ma wartość 4, prawo SGID wartość 2, prawo SVTX wartość 1, prawo odczytu wartość 4, prawo zapisu wartość 2 i prawo wykonywania wartość 1. Jeśli dane prawo nie jest nadane, to ma wartość 0. Następnie w obrębie każdej trójki dokonujemy sumowania i otrzymana cyfra oznacza prawo dostępu. Przykładowo wartość praw dostępu 2754 oznacza, że: ustawione jest prawo SGID (2), właściciel indywidualny ma prawo odczytu, zapisu i wykonywania ( $4 + 2 + 1 = 7$ ), właściciel grupowy ma prawo odczytu i wykonania ( $4 + 1 = 5$ ), natomiast pozostali użytkownicy mają prawo odczytu (4). Wracając do przykładowej umaski, umożliwia ona nadawanie wszystkich praw dodatkowych, prawa odczytu, zapisu i wykonywania właścicielowi indywidualnemu, prawa odczytu i wykonywania właścicielowi grupowemu (gdyż wartość 2 oznacza cofnięcie prawa zapisu) oraz prawa odczytu i wykonywania pozostałym użytkownikom systemu. Stąd utworzone plik oraz katalog mają prawa dostępu takie jak przedstawiono poniżej:

```
1 -bash-4.0$ touch plik; mkdir katalog
2 -bash-4.0$ ls -ld plik katalog
3 -rw-r--r--  1 wacek  1001      0 Jun 24 17:18 plik
4 drwxr-xr-x  2 wacek  1001    512 Jun 19 13:51 katalog
```

Plik regularny oraz katalog mają cofnięte prawa zapisu dla właściciela grupowego oraz pozostałych użytkowników systemu zgodnie z wartością umaski. Dodatkowo plik regularny ma cofnięte prawa wykonywania.

Wartość maski ma wpływ na bezpieczeństwo systemu. Maską 0022 jest liberalną. W systemach przeznaczonych dla dostępu szerokiej społeczności użytkowników stosuje się maskę 0077, która pozwala przydzielać prawa dostępu jedynie właścicielowi indywidualnemu.

W omawianym modelu zarządzanie prawami dostępu do obiektu jest przekazane jego właścicielowi. Może on dokonywać ich modyfikacji przy pomocy komendy *chmod*. Komenda ta wymaga podania sposobu zmiany lub docelowych praw dostępu oraz, jako argumentu, listy nazw plików, których prawa dostępu mają zostać zmienione. Modyfikacji praw dostępu przez podanie zmian w prawach istniejących stosuje się jeśli, modyfikacje te są niewielkie. Specyfikacja ta składa się z trzech fragmentów:

1. Której trójki praw modyfikacja dotyczy: **u** – właściciel indywidualny, **g** – właściciel grupowy, **o** – pozostali użytkownicy systemu, **a** – modyfikacje dotyczyć będą wówczas wszystkich wymienionych.
2. Sposobu modyfikacji: **+** – oznacza nadanie prawa, **-** – cofnięcie prawa, **=** – cofnięcie wszystkich istniejących i nadanie wyspecyfikowanych.
3. Które prawo ma zostać zmodyfikowane: **r** – prawo odczytu, **w** – prawo zapisu, **x** – prawo wykonywania, **s** – SUID lub SGID w zależności od tego, czy dotyczy właściciela indywidualnego czy grupowego, **t** – SVTX.

Posłużmy się następującym przykładem. Dany jest katalog o prawach dostępu jak poniżej:

```
1 drwxr-xr-x 2 antek pracownicy 512 Jun 25 18:09 katalog01
```

Prawa dostępu modyfikujemy następującą komendą:

```
1 $ chmod g=w,o+w,+t katalog01
```

Wyspecyfikowane modyfikacje wymagają cofnięcia wszystkich praw właścicielowi grupowemu i nadanie mu jedynie prawa zapisu, dodanie prawa zapisu pozostałym użytkownikom systemu oraz nadanie prawa SVTX (nadanie prawa SVTX nie wymaga podania właściciela. W przypadku prawa SUID musimy wskazać na właściciela indywidualnego: *u+s*, zaś ustawiając prawo SGID na właściciela grupowego: *g+s*). Specyfikacje zmian praw dostępu stanowią jeden ciąg znaków i są oddzielone przecinkiem. Nie może się tu pojawić jakiegokolwiek znak biały. Po wykonaniu komendy katalog otrzymuje prawa dostępu jak poniżej:

```
1 $ ls -ld katalog01
2 drwx-w-rwt 2 antek 1003 512 Jun 25 20:07 katalog01
```

Specyfikowanie zmina w prawach dostępu jest wygodne, jeśli zmiany te są kosmetyczne: dodajemy jedno prawo, usuwamy drugie. Jeśli zmiana ma być radykalna, to często lepiej jest posłużyć się zapisem ósemkowym i wyspecyfikować docelowe prawa dostępu. Przykładowo:

```
1 $ chmod 754 katalog01
```

Specyfikacja praw dostępu składa się z trzech cyfr opisujących docelowe prawa dostępu, kolejno od lewej dla właściciela indywidualnego, grupowego i pozostałych użytkowników systemu. W tym przypadku właściciel indywidualny otrzyma prawo odczytu, zapisu i wykonywania ( $4 + 2 + 1 = 7$ ), właściciel grupowy prawo odczytu i wykonywania ( $4 + 1 = 5$ ), zaś pozostali użytkownicy systemu jedynie prawo odczytu (4). Wykonanie komendy spowoduje cofnięcie praw istniejących i nadanie wyspecyfikowanych:

```
1 drwxr-xr-- 2 antek 1003 512 Jun 25 20:07 katalog01
```

Jeśli wymagane jest ustawienie rozszerzonych praw dostępu, to specyfikacja składa się z czterech cyfr, a prawa te reprezentuje skrajna, lewa pozycja. Przykładowo:

```
1 $ chmod 3770 katalog01
```

Wykonanie powyższej komendy spowoduje ustawienie prawa SGID oraz SVTX ( $2 + 1 = 3$ ), oraz wszystkich praw dostępu właścicielowi indywidualnemu (7) i grupowemu (7). Pozostałym użytkownikom systemu nie ustawi żadnych praw (0). Duża litera T w poniższym opisie informuje, że prawo wykonywania dla pozostałych użytkowników systemu nie jest ustawione.

```
1 drwxrws--T 2 antek 1003 512 Jun 25 20:07 katalog01
```

Komenda *chmod* posiada kilka użytecznych opcji. W dystrybucji Fedora należy do nich zaliczyć:

- **-c** – raportuje zmiany praw dostępu, które zostały wykonane.
- **-f** – użycie opcji powoduje wypisywanie komunikatów o najistotniejszych błędach.
- **-v** – wypisuje informacje o rozpoczęciu zmian praw dostępu każdego pliku.
- **-R** – użycie opcji umożliwia rekursywną zmianę praw dostępu, od katalogu wskazanego jako argument do końca drzewa katalogów.
- **-reference=ścieżka\_dostępu\_do\_pliku** – umożliwia nadanie praw dostępu takich jakie posiada plik, którego nazwa pojawiła się jako wartość opcji.

W dystrybucji FreeBSD należy wskazać na następujące opcje:

- **-f** – użycie tej opcji powoduje, że nie są raportowane błędy np. o zakończonych niepowodzeniem próbach zmiany praw dostępu.
- **-h** – używana jeśli zmiana praw dostępu dotyczy pliku będącego dowiązaniem symbolicznym. Powoduje, że zmiana dotyczy pliku będącego dowiązaniem, a nie pliku na który dowiązanie to wskazuje.
- **-R** – opcja umożliwia zmianę praw dostępu rekursywnie, począwszy od katalogu do którego ścieżka dostępu pojawiła się jako argument wywołania komendy, do końca drzewa katalogów. Jeśli jednak w katalogu występują pliki będące dowiązaniami symbolicznymi, to o zmianie praw dostępu plików, na które wskazują decyduje użycie poniższych opcji **-L**, **-P** oraz **-H**.

- **-L** – opcja używana z opcją **-R**. Jeśli w katalogu, w którym prawa dostępu są zmieniane występują dowiązania symboliczne, to wszystkim plikom na które one wskazują zostaną zmienione prawa dostępu.
- **-P** – opcja używana z opcją **-R**. Jeśli w katalogu, w którym prawa dostępu są zmieniane występują dowiązania symboliczne, to żadnemu plikowi na który one wskazują nie zostaną zmienione prawa dostępu. Jest to opcja domyślna.
- **-H** – opcja używana z opcją **-R**. Jeśli w katalogu, w którym prawa dostępu są zmieniane występują dowiązania symboliczne, to prawa dostępu zostaną zmienione jedynie tym plikom, do których ścieżki pojawiły się jako argumenty wywołania komendy.
- **-v** – użycie tej opcji wymusza tryb gadatliwy. W praktyce raportowane są informacje o każdej podjętej próbie zmiany praw dostępu, błędach wynikłych podczas wykonywania tej operacji, itd.

### Listy kontroli dostępu

Zaletą przedstawionego powyżej mechanizmu kontroli dostępu jest prostota i przejrzystość. Jego istotną wadą jest natomiast coś, co można określić pojęciem *gruboziańskości*. Wynika ona z gradacji właścicieli. Prawa dostępu dotyczą właściciela indywidualnego, użytkowników należących do grupy będącej właścicielem grupowym oraz wszystkich pozostałych użytkowników systemu. Przykładowo, jeśli jesteśmy właścicielem pliku i chcemy innemu użytkownikowi umożliwić modyfikowanie jego zawartości, to należy to zrobić wykorzystując prawa dostępu dla właściciela grupowego. Do nas należy ustawienie odpowiednich praw dostępu, zaś do administratora systemu utworzenie grupy, a następnie przypisanie użytkownika do grupy będącej właścicielem grupowym pliku. Jak widać prowadzi to do wzrostu liczby grup w systemie oraz wzrostu liczby grup, do których należy dany użytkownik. Możliwości systemu, zwłaszcza w drugim przypadku są ograniczone. Współdzielenie natomiast w oparciu o prawa dostępu dla pozostałych użytkowników systemu jest po prostu niebezpieczne.

Wady tradycyjnego systemu praw dostępu wymusiły powstanie mechanizmu znacznie bardziej elastycznego, a mianowicie list kontroli dostępu (ang. *Access Control List*, ACL). W chwili obecnej są one dostępne w niemal wszystkich uniksowych i linuksowych systemach plików, w systemie NTFS jak również w sieciowych systemach plików, np. w *Andrew File System* (AFS). Definiuje je standard POSIX 1003.1e draft 17 zwany krótko POSIX.1e.

Zacznijmy od teorii. Prawa dostępu do obiektu (pliku lub procesu) zdefiniowane są za pomocą tzw. listy wpisów. Trzy pierwsze wpisy definiują prawa dostępu odpowiednio dla właściciela indywidualnego, właściciela grupowego i pozostałych użytkowników systemu. Stanowią odpowiednik tradycyjnego systemu praw dostępu i są nazywane minimalną listą kontroli dostępu (ang. *minimal ACLs*). Lista związana z danym obiektem, posiadająca więcej niż trzy wpisy nazywana jest listą rozszerzoną (ang. *extended ACLs*). Wpisy te mogą definiować maskę oraz prawa dostępu dla tzw. użytkownika nazwanego oraz grupy nazwanej. Typy lub klasy wpisów oraz ich oznaczenia symboliczne przedstawiono w tabeli 3.1.

Wpisy typu *użytkownik nazwany* oraz *grupa nazwana* określa się mianem *klasy grupowej*. W przypadku listy minimalnej, prawa dostępu dla klasy grupowej odpowiadają prawom dostępu dla właściciela grupowego. W listach rozszerzonych zawarte są dodatkowe wpisy definiujące prawa dostępu dla innych użytkowników i grup. Rodzi to pewne problemy, gdyż zarówno dodatkowe wpisy mogą zawierać definicję praw dostępu, których nie posiada właściciel grupowy, jak również prawa dostępu właściciela grupowego mogą być sprzeczne z tymi zdefiniowanymi dla klasy grupowej. Rozwiązaniem problemu jest wprowadzenie tzw. wpisu maski (ang. *mask entry*). W

Typ wpisu	Oznaczenie symboliczne
Właściciel indywidualny	<code>user::rwx</code>
Użytkownik nazwany	<code>user:<i>nazwa</i>:rwx</code>
Właściciel grupowy	<code>group::rwx</code>
Grupa nazwana	<code>group:<i>nazwa</i>:rwx</code>
Maska	<code>mask::rwx</code>
Pozostali użytkownicy	<code>other::rwx</code>

Tablica 3.1: Typy wpisów ACL oraz ich oznaczenia symboliczne wykorzystywane do zarządzania prawami dostępu.

listach minimalnych prawa dostępu z wpisu dla właściciela grupowego są bezpośrednio mapowane na prawa dostępu dla właściciela grupowego. W listach rozszerzonych wypisywane prawa dostępu dla właściciela grupowego należy odczytywać jako maskę, zaś prawa dostępu dla właściciela grupowego są zapisane we wpisie typu grupa. Domyślnie prawa dostępu dla właściciela grupowego i maska są jednakowe. Sposób interpretacji minimalnej i rozszerzonej listy kontroli dostępu przedstawiono na rysunku 3.5a.

Zdefiniowana maska mówi jakie efektywne (maskymalne) prawa dostępu są dostępne w obrębie wpisów klasy grupowej. W przypadku listy minimalnej są one zapisane wprost. Określenie praw dostępu dla wpisu z klasy grupowej wymaga przyłożenia do praw danego wpisu maski i uznanie jedynie tych praw, które zostały ustalone w masce. Przestawia to rysunek, w którym za przykład posłużył wpis dla użytkownika *antek* z rysunku 3.5b. Jak widać, maska dopuszcza jedynie prawa odczytu i zapisu. Prawo wykonywania jest cofnięte. Użytkownikowi *antek* nadano prawa odczytu i wykonywania. Natomiast po uwzględnieniu maski efektywne prawa użytkownika zostają ograniczone do prawa odczytu.

Generalnie rozróżnia się dwa typy list kontroli dostępu. Są to: lista *dostępowa* (ang. *access ACL*) oraz *domyślna* (ang. *default ACL*). Pierwsza z nich opisuje prawa dostępu do istniejącego obiektu, druga zaś prawa dostępu dziedziczone przez obiekt w momencie jego tworzenia w systemie plików po katalogu nadrzędnym. Domyślne listy kontroli dostępu mają praktyczne znaczenie jedynie dla katalogów. Dla plików nie będących katalogami są one bez znaczenia, gdyż nie można utworzyć w nich innych plików. Stąd nie musi on dysponować listami kontroli dostępu do przekazania tworzonemu plikowi. Jeśli w danym katalogu zostanie utworzony katalog, to dziedziczy on po katalogu nadrzędnym zarówno *dostępową*, jak i *domyślną* listę kontroli dostępu. Jeśli natomiast tworzony jest obiekt nie będący katalogiem, to po katalogu nadrzędnym dziedziczy jedynie listę *dostępową*. Odziedziczone prawa dostępu są następnie modyfikowane w momencie tworzenia danego pliku w wyniku wywołania funkcji systemowej *mode*. Jeśli katalog nadrzędny nie posiada zdefiniowanych domyślnych list kontroli dostępu, to są one określane z wykorzystaniem algorytmu opisanego w standardzie POSIX.1. Istotnym jest, że wartość umaski nie ma wpływu na prawa dostępu jeśli w katalogu zdefiniowana została domyślna lista kontroli dostępu.

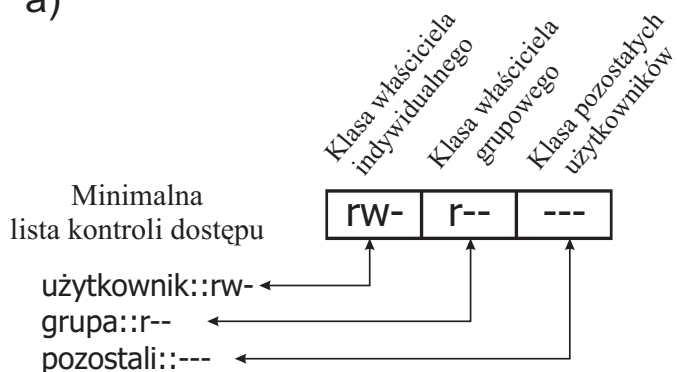
Zarządzanie listami kontroli dostępu to praktycznie trzy komendy: *getfacl*, *setfacl* oraz w niektórych systemach *chacl*. Komenda *getfacl* służy do wypisywania list kontroli dostępu obiektów, których nazwy pojawiły się jako argumenty wywołania. Przykładowo, w katalogu bieżącym tworzymy plik regularny oraz katalog i listujemy ich listy kontroli dostępu. Wynik przedstawiono poniżej:

```

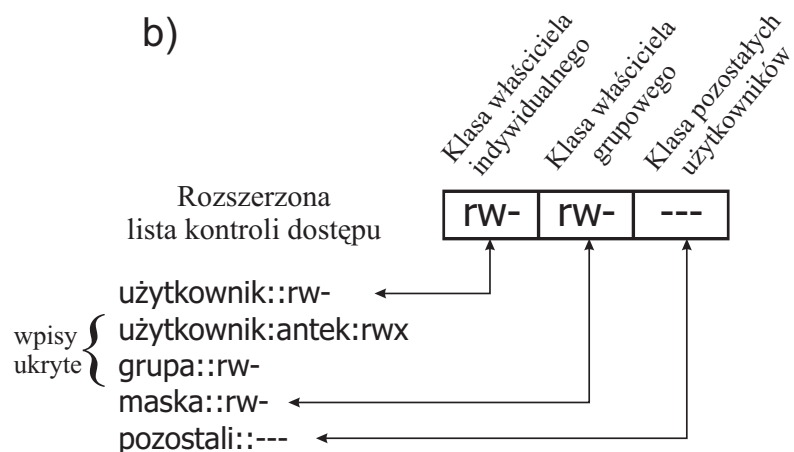
1 [antek@messy Katalog]$ mkdir katalog01
2 [antek@messy Katalog]$ touch plik01
3 [antek@messy Katalog]$ getfacl plik01 katalog01
4 # file: plik01
5 # owner: antek

```

a)



b)



Rysunek 3.5: Interpretacja praw dostępu dla a) minimalnej oraz b) rozszerzonej listy kontroli dostępu.

```

6  # group: pracownicy
7  user::rw-
8  group::r--
9  other::r--
10
11 # file: katalog01
12 # owner: antek
13 # group: pracownicy
14 user::rwx
15 group::r-x
16 other::r-x

```

Jak widać oba obiekty posiadają minimalne listy kontroli dostępu. Właścicielem indywidualnym jest użytkownik *antek*, zaś grupowym jego grupa podstawowa również o nazwie *antek*.

Założmy, że użytkownik *jan* powinien mieć prawo odczytu oraz zapisu pliku *plik01* oraz pełne prawa dostępu do katalogu *katalog01*. Dodatkowo użytkownik *wacek* powinien mieć pełne prawa dostępu zarówno do pliku *plik01* jak również katalogu *katalog01*. Prawa obowiązujące dla katalogu *katalog01* powinny być dziedziczone przez każdy plik użytkownika *wacek* utworzony w tym katalogu. Powinien on zatem posiadać domyślne listy kontroli dostępu.

Wykorzystując możliwości klasycznych praw dostępu zadanie to jest kłopotliwe do zrealizowania. Rozwiązanie polegałoby na stworzeniu grupy użytkowników oraz uczynieniu jej dodatkową dla użytkowników *jan* oraz *wacek*. Następnie grupę tę należy uczynić właścicielem grupowym pliku, do którego dostęp ma być współdzielony przez jej członków. I tu pojawia się problem, gdyż hipotetycznie w grupie mogą znaleźć się użytkownicy, którzy powinni posiadać różne prawa dostępu. A jak wiemy właściciel grupowy jest jeden i posiada on ściśle określone prawa dostępu. Dodatkowo, przy takim rozwiązaniu zmiana praw dostępu wiązałaby się teoretycznie z dopisaniem użytkownika, którego prawa dostępu ulegają modyfikacji do odpowiedniej grupy. Zwróćmy uwagę, że wszystkie te czynności wymagają zaangażowania użytkownika *root* lub administratora grupy.

Użycie list kontroli dostępu upraszcza administrowanie prawami dostępu angażując do tego, zgodnie z zasadą modelu DAC, jedynie użytkownika będącego właścicielem obiektu, który wykorzystuje w tym celu komendę *setfacl*. Realizacja przedstawionego powyżej przykładu wygląda następująco:

```
1 [antek@messy Katalog]$ setfacl -m u:jan:rw plik01
2 [antek@messy Katalog]$ setfacl -m u:jan:rwk katalog01
3 [antek@messy Katalog]$ setfacl -m u:wacek:rwk plik01
4 [antek@messy Katalog]$ setfacl -m d:u:wacek:rwk katalog01
```

Dla ułatwienia, tworzenie każdego wpisu do list kontroli dostępu zrealizowano oddzielną komendą. Po ich wykonaniu listy kontroli dostępu są następujące:

```
1 [antek@messy Katalog]$ getfacl plik01 katalog01
2 # file: plik01
3 # owner: antek
4 # group: pracownicy
5 user::rw-
6 user:wacek:rwk          #effective:r-x
7 user:jan:rw-            #effective:r--
8 group::r--
9 mask::r-x
10 other::r--
11
12 # file: katalog01
13 # owner: antek
14 # group: pracownicy
15 user::rwk
16 user:jan:rwk            #effective:r-x
17 group::r-x
18 mask::r-x
19 other::r-x
20 default:user::rwk
21 default:user:wacek:rwk
```

```

22 default:group::r-x
23 default:mask::rwx
24 default:other::r-x

```

Zwróćmy uwagę, iż na listach kontroli dostępu pliku *plik01* pojawiły się trzy dodatkowe wpisy. Wpis *mask*, jak wspomniano, definiuje efektywne prawa dostępu do obiektu. Są to prawa odczytu i wykonywania. Pozostałe dwa to wpisy definiujące prawa dostępu dla użytkowników. I tak użytkownik *wacek* ma prawa odczytu, zapisu i wykonywania, ale po uwzględnieniu maski pozostają prawa odczytu i wykonywania. Użytkownik *jan* posiada prawa odczytu i zapisu, ale efektywnie jedynie prawo odczytu. W przypadku katalogu *katalog01* sytuacja wygląda nieco inaczej. Pojawiła się mianowicie tzw. *domyślna* lista kontroli dostępu, na której wpisy oznaczone są jako *default*. Wpisy te definiują prawa dostępu dla plików tworzonych w tym katalogu przez jego właściciela indywidualnego (wpis *default:user::rwx*), właściciela grupowego (wpis *default:group::r-x*), pozostałych użytkowników systemu (wpis *default:other::r-x*) oraz maskę (wpis *default:mask::rwx*). Dodatkowo zostały zdefiniowane prawa dostępu do plików tworzonych przez użytkownika *wacek* (wpis *default:user:wacek:rwx*). Będą one obowiązywały od bieżącego katalogu do końca drzewa katalogów chyba, że w pewnym miejscu zostaną one usunięte. Wpis ten jest wynikiem użycia przełącznika **d** w wywołaniu komendy *setfacl*. Na liście kontroli dostępu katalogu *katalog01* pojawił się także wpis *dostępowy* informujący, że użytkownik *jan* posiada prawo odczytu, zapisu i wykonywania zaś efektywnie prawo odczytu i wykonywania.

Pliki posiadające listy kontroli dostępu oznaczane są znakiem **+** w polu opisującym prawa dostępu. W naszym przypadku:

```

1 [antek@messy Katalog]$ ls -l
2 total 4
3 drwxr-xr-x+ 2 antek pracownicy 4096 2009-07-15 16:16 katalog01
4 -rw-r-xr--+ 1 antek pracownicy    0 2009-07-15 16:16 plik01

```

W przypadku plików oznaczonych w ten sposób należy pamiętać, iż prawa dostępu dla właściciela grupowego oznaczają w praktyce maskę, zaś obowiązujące prawa dostępu można uzyskać komendą *getfacl*.

Trzecią komendą umożliwiającą zarządzanie listami kontroli dostępu jest komenda *chacl*. Pozwala ona na modyfikowanie list kontroli dostępu do pliku lub katalogu. Jak wspomniano, nie jest ona dostępna we wszystkich systemach tej rodziny. Jej implementacje są dostępne w systemach linuksowych i niektórych systemach komercyjnych. Nie występuje w systemach BSD.

Użycie polecenia *chacl* wymaga wyspecyfikowania sposobu modyfikacji listy ACL, poprzez podanie nowej postaci jej właściwego fragmentu oraz jako argumentów nazw plików i katalogów, których ta zmiana dotyczy. Stąd ogólna postać, z uwzględnieniem ewentualnych opcji, jest następująca: *chacl [opcje] acl nazwa\_pliku ...*. Napis oznaczony symbolem *acl* jest interpretowany przez funkcję *acl\_from\_text* i składa się z oddzielonych przecinkami klauzul o następującej postaci: *tag:nazwa:prawo\_dostepu*. *tag* może mieć jedną z następujących postaci:

- *user* lub *u* – umożliwia wskazanie nazwy właściciela indywidualnego we wpisach ACL.
- *group* lub *g* – umożliwia wskazanie nazwy właściciela grupowego we wpisach ACL.
- *other* lub *o* – specyfikuje wpisy dotyczące pozostałych użytkowników w systemie.
- *mask* lub *m* – oznacza modyfikację maski.



*nazwa* specyfikuje nazwę użytkownika lub grupy zdefiniowanej w systemie. Jeśli jest ona napisem pustym, to domniemuje się, w zależności od kontekstu, właściciela indywidualnego lub grupowego. *prawo\_dostępu* są napisem w postaci *rwX*, gdzie każde z liter reprezentująca prawo dostępu może być zastąpione znakiem *-*, oznaczającym oczywiście brak jego występowania.

Jeśli plik lub katalog posiadają jedynie minimalną listę kontroli dostępu, to specyfikuje się zmiany jej dotyczące. Przykładowo:

```
1 antek@messy Katalog]$ chmod u::rw-,g::r--,o::r-- plik01
```

W pozostałych przypadkach wymagane jest podanie sposobu modyfikacji w zakresie obowiązującym właścicieli indywidualnego, grupowego i pozostałych użytkowników systemu oraz modyfikacji dotyczących wybranych wpisów listy kontroli dostępu. Przykład przedstawiono poniżej.

```
1 [antek@messy Katalog]$ getfacl plik01
2 # file: plik01
3 # owner: antek
4 # group: pracownicy
5 user::rw-
6 user:jan:rw-          #effective:r--
7 user:wacek:rwX        #effective:r-x
8 group::r--
9 mask::r-x
10 other::r--
11
12 [antek@messy Katalog]$ chmod u::rw-,g::r--,o::r--u:jan:r--,u:wacek:r--,m::rwX plik01
13
14 [antek@messy Katalog]$ getfacl plik01
15 # file: plik01
16 # owner: antek
17 # group: pracownicy
18 user::rw-
19 user:jan:r--
20 user:wacek:r--
21 group::r--
22 mask::rwX
23 other::r--
```

Polecenie *chmod* posiada kilka opcji, z których najczęściej używanymi są:

- **-b** – opcja informuje, że do zmiany są dwie listy ACL. Pierwsza z nich jest listą ACL dla pliku, druga domyślną listą ACL dla katalogu.
- **-d** – umożliwia ustawienie wartości domyślnej listy ACL dla katalogu.
- **-R** – opcja pozwala usunąć jedynie listę ACL dla pliku.
- **-D** – opcja pozwala usunąć jedynie listę ACL dla katalogu.
- **-B** – umożliwia usunięcie całej listy ACL.

- **-l** – opcja umożliwia wypisanie zawartości listy ACL oraz domyślnej listy ACL wskazanego pliku lub katalogu. Opcja ta nie występuje we wszystkich implementacjach polecenia *chacl*.
- **-r** – pozwala na rekurencyjne nadanie wartości listom ACL od katalogu wskazanego przez argument polecenie *chacl*. Opcja ta nie jest dostępna we wszystkich jego implementacjach.

Listy kontroli dostępu stanowią swoistą nakładkę na klasyczny mechanizm praw dostępu występujący w systemie Unix. Jak w takim razie rozstrzygać sytuacje wątpliwe, w których prawa określone mechanizmem tradycyjnym są różne od tych opisanych listą kontroli dostępu. Pomocnym może być przedstawiony poniżej algorytm 1.

---

**Algorithm 1** Algorytm weryfikacji dostępu dla list kontroli dostępu.

---

```

if Numer identyfikacyjny użytkownika będącego właścicielem procesu jest identyczny z nume-
    rem identyfikacyjnym właściciela pliku then
    obowiązuja prawa dostępu właściciela indywidualnego.
else if Numer identyfikacyjny użytkownika będącego właścicielem procesu odpowiada nume-
    rowi identyfikacyjnemu jednego z wpisów nazwanych użytkownika then
    prawa określone w tym wpisie determinują dostęp.
else if Którykolwiek numer identyfikacyjny grupy będącej właścicielem procesu jest identyczny
    z numerem identyfikacyjnym właściciela grupowego i prawa w nim określone pozwalają na
    wykonanie żądanej operacji then
    wpis ten określa prawa dostępu.
else if Którykolwiek numer identyfikacyjny grupy będącej właścicielem procesu odpowiada
    numerowi identyfikacyjnemu jednego z wpisów nazwanych grupy i prawa w nim określone
    pozwalają na wykonanie żądanej operacji then
    wpis ten określa prawa dostępu.
else if Którykolwiek numer identyfikacyjny grupy będącej właścicielem procesu odpowiada
    numerowi identyfikacyjnemu właściciela grupowego ale nie odpowiada mu żaden z wpisów
    nazwanych grupy jak również prawa dostępu dla właściciela grupowego oraz prawa dostępu
    z żadnego odpowiadającego wpisu nazwanego grupy nie pozwalają na wykonanie żądanych
    operacji then
    dostęp jest zabroniony.
else
    Inny wpis określa prawa dostępu
end if
if Wpis otrzymany algorytmem dopasowania odpowiada wpisowi dla właściciela indywidual-
    nego lub innemu wpisowi then
    dostęp zostaje przyznany.
else if Otrzymany wpis jest wpiasem nazwanym użytkownika, wpisem właściciela grupowego,
    lub wpisem nazwanym grupy i wpis ten zawiera żądane efektywne prawa dostępu then
    dostęp zostaje przyznany.
else
    Dostęp jest zabroniony.
end if

```

---

Jak widać algorytm jest zagmatwany. Upraszczając go, sprawdzanie praw dostępu zaczynamy od praw tradycyjnych, a następnie przeglądamy odpowiednie wpisy nazwane uwzględniając zdefiniowaną maskę.

Komendy pozwalające na zarządzanie listami kontroli dostępu zostały przedstawione w przykładach. Omówmy ich składnię oraz podstawowe opcje. Zaczniemy od komendy *getfacl*. Zarówno

w dystrybucjach RedHat jak i FreeBSD komendę tę można użyć bez opcji podając jako argument listę plików, których listy kontroli dostępu mają zostać wypisane. Do podstawowych opcji komendy w dystrybucji RedHat należą:

- **-a** – opcja domyślna – wypisuje listy kontroli dostępu.
- **-c** – opcja powoduje pominięcie nagłówka informującego o nazwie pliku, jego właścicielu indywidualnym oraz grupowym (po prostu komentarzy, czyli linii zaczynających się znakiem #).
- **-d** – użycie opcji powoduje wypisanie jedynie domniemanych list kontroli dostępu (istotnych w przypadku katalogów).
- **-e** – opcja umożliwia wypisanie wszystkich efektywnych praw dostępu nawet jeśli są one identyczne z umieszczonymi we wpisach. Pominięcie tej opcji powoduje wypisanie jedynie tych efektywnych praw dostępu, które z powodu zdefiniowanej maski różnią się od określonych we wpisach.
- **-E** – użycie opcji spowoduje, iż żadne efektywne prawa dostępu nie zostaną wypisane.
- **-s** – opcja spowoduje, iż nie pojawi się informacja o plikach posiadających jedynie minimalną listę kontroli dostępu.
- **-R** – opcja umożliwia wypisywanie list kontroli dostępu w sposób rekursywny, czyli od katalogu bieżącego lub wskazanego w argumencie wywołania komendy do końca drzewa katalogów.
- **-L** – opcja działa efektywnie z opcją **-R**. Umożliwia wypisywanie list kontroli dostępu do plików znajdujących się w katalogach, które wskazywane są przez dowiązania symboliczne (komenda „podąża” za dowiązaniem symbolicznym).
- **-P** – opcja ta powoduje działanie odwrotne do opcji **-L**, uniemożliwiając wypisywanie list kontroli dostępu plików znajdujących się w katalogu wskazywanym przez dowiązanie symboliczne. Działa efektywnie z opcją **-R**.
- **-t** – użycie opcji umożliwia uzyskanie alternatywnego formatu informacji o listach kontroli dostępu. Ma on postać tabeli, którą dla naszego przykładu przedstawiono poniżej. Dużymi literami zostały zaznaczone prawa usunięte przez maskę.

```

1 [antek@messy Katalog]$ getfacl -t *
2 # file: katalog01
3 USER antek      rwx  rwx
4 user wacek      rwx
5 user jan        rWx
6 GROUP pracownicy r-x  r-x
7 mask           r-x  rwx
8 other          r-x  r-x
9
10 # file: plik01
11 USER antek      rw-
12 user wacek      rWx
13 user jan        rW-
14 GROUP pracownicy r--

```

15	<b>mask</b>	<b>r-x</b>
16	<b>other</b>	<b>r--</b>

- **-p** – użycie opcji spowoduje, że ze ścieżek dostępu do plików, których zawartość listy kontroli dostępu jest wypisywana poniżej nie będą usunięte rozpoczynające je znaki ./.
- **-n** – w wyniku użycia opcji zamiast nazw użytkowników i grup, których wpisy dotyczą pojawiają się ich numery identyfikacyjne.

W systemie FreeBSD komenda *getfacl* posiada następujące opcje:

- **-d** – użycie opcji powoduje, iż komenda *getfacl* wypisuje jedynie domyślne wpisy z listy kontroli dostępu. Jeśli plik ich nie posiada komenda zwraca błąd.
- **-h** – jeśli argumentem komendy jest dowiązanie symboliczne, to użycie opcji powoduje, że wypisywana jest lista kontroli dostępu pliku będącego dowiązaniem, a nie pliku na który dowiązanie to pokazuje.
- **-q** – użycie opcji powoduje, że nie są wypisywane informacje o nazwie pliku oraz jego właścicielu indywidualnym i grupowym, czyli wszystkie linie rozpoczynające się znakiem #.

Składnia komendy *setfacl* jest nieco bardziej skomplikowana. Oczywiście wymaga ona podania argumentu w postaci listy nazw plików, których listy kontroli dostępu mają zostać zmodyfikowane. Konieczne jest również podanie wpisu listy kontroli dostępu, który występuje po opcji **-m** lub **-x**. Opcja **-m** służy do wstawiania wpisu, zaś **-x** do usuwania wpisu z listy kontroli dostępu. Definicja wpisu może przyjmować jedną z ogólnych postaci:

1. **[d[efault]:] u[ser]:uid [:perms]** – pozwala zdefiniować wpis dla użytkownika o podanej nazwie lub numerze identyfikacyjnym, a dla właściciela indywidualnego jeśli wartość ta zostanie pominięta. Przykładowo: *u:jan:rw* ustala dostępową listę kontroli dostępu dla użytkownika *jan* z prawami odczytu, zapisu i wykonywania, zaś *d:u::r*— pozwala zdefiniować domniemaną listę kontroli dostępu dla właściciela indywidualnego z prawem odczytu.
2. **[d[efault]:] g[roup]:gid [:perms]** – pozwala zdefiniować wpis dla grupy o podanej nazwie lub numerze identyfikacyjnym, a dla właściciela grupowego jeśli wartość ta zostanie pominięta. Przykładowo: *g:513:rw*— definiuje dostępową listę kontroli dostępu dla grupy o numerze identyfikacyjnym GID 513 z prawami odczytu i zapisu, zaś *default:g:pracownicy:rw*— określa domyślną listę kontroli dostępu dla grupy *pracownicy* z prawami odczytu i zapisu.
3. **[d[efault]:] m[ask][:] [:perms]** – służy do definiowania maski. Przykładowo: *m::rx* - definiuje maskę dla wpisów typu użytkownik nazwany i grupa nazwana z dopuszczeniem praw zapisu i wykonania. Praktycznie oznacza to cofnięcie prawa zapisu dla wszystkich wpisów typu użytkownik nazwany i grupa nazwana.
4. **[d[efault]:] o[ther][:] [:perms]** – umożliwia definiowanie wpisów dla wszystkich pozostałych. Przykładowo wpis: *d:o::400* definiuje domyślną listę kontroli dostępu dla pozostałych użytkowników jedynie z prawem odczytu.

Komenda *setfacl* posiada również opcje. W dystrybucji RedHat do najczęściej wykorzystywanych należą:

- **-b** – użycie opcji powoduje usunięcie wszystkich rozszerzonych wpisów z listy kontroli dostępu.
- **-k** – opcja służy do usuwania wpisów domyślnych z listy kontroli dostępu. Jeśli nie występują, komenda nie wypisuje żadnego komunikatu.
- **-n** – opcja powoduje, że efektywne prawa dostępu nie zostaną wyliczone. Domyślne działanie komendy *setfacl* wymusza przeliczanie efektywnych praw dostępu, jeśli zdefiniowana została przy jej użyciu nowa maska. Wyznaczenie efektywnych praw dostępu można wymusić używając komendy *setfacl* z opcją **-mask**
- **-d** – użycie opcji powoduje, że wyspecyfikowane zmiany praw dostępu będą dotyczyły wpisów domyślnych.
- **-R** – komenda uruchomiona z tą opcją działa rekursywnie ustalając wpisy na listach kontroli dostępu wszystkim plikom od katalogu bieżącego lub tego, którego nazwa pojawiła się jako argument do końca drzewa katalogów.
- **-L** – opcja działa efektywnie w połączeniu z opcją **-R** i wymusza zmiany wpisów na listach kontroli dostępu w katalogach, do których dowiązanie symboliczne pojawiło się w bieżącym katalogu.
- **-P** – wymusza działanie odwrotne do zdefiniowanego opcją **-P**, a mianowicie jeśli w bieżącym katalogu pojawiło się dowiązanie symboliczne wskazujące na katalog, to jej użycie powoduje, iż nie są modyfikowane wpisy z list kontroli dostępu znajdujących się w nim plików. Opcja działa efektywnie jedynie z opcją **-R**.
- **-** – służy do zaznaczenia końca listy opcji. Wszystkie występujące po nim napisy są traktowane jako nazwy plików, nawet jeśli zostały poprzedzone znakiem **-**.
- **-** – jeśli zamiast nazw plików, których wpisy list kontroli dostępu mają zostać zmodyfikowane pojawi się znak **-**, to komenda *setfacl* czyta je ze standardowego wejścia.
- **-M ścieżka\_dostępu\_do\_pliku** – opcja służy do modyfikowania istniejących lub tworzenia nowych wpisów w listach kontroli dostępu zgodnie z opisem zawartym w pliku, ścieżka dostępu do którego jest wartością opcji. Jeśli zamiast nazwy pliku pojawił się znak **-**, to komenda czyta dane ze standardowego wejścia.
- **-X ścieżka\_dostępu\_do\_pliku** – opcja umożliwia usuwanie wpisów z list kontroli dostępu zgodnie z opisem zawartym w pliku, ścieżka dostępu do którego jest wartością opcji.

W dystrybucji FreeBSD do najczęściej wykorzystywanych opcji komendy *setfacl* należą:

- **-b** – użycie tej opcji skutkuje usunięciem wszystkich wpisów tworzących rozszerzoną listę kontroli dostępu. Usunięcie wpisu typu maska powoduje, iż prawa dostępu dla właściciela grupowego przejmują jej funkcję.
- **-d** – opcja powoduje, iż operacje zmiany lub usunięcia wpisów będą dotyczyły jedynie domyślnej listy kontroli dostępu. Lista domniemana nie zostanie zmieniona.
- **-h** – efektywnie opcja działa w stosunku do plików będących dowiązaniami symbolicznymi modyfikując ich listy kontroli dostępu, a nie plików na które dowiązania te pokazują. Domyślnie modyfikowane są listy kontroli dostępu plików na które dowiązania pokazują.

- **-k** – opcja umożliwia usunięcie każdego wpisu z listy kontroli dostępu.
- **-n** – opcja służy do zabronienia wyznaczanie efektywnych praw dostępu w przypadku zmiany wpisów lub maski.
- **-M ścieżka\_dostępu\_do\_pliku** – opcja służy do modyfikowania istniejących lub tworzenia nowych wpisów w listach kontroli dostępu zgodnie z opisem zawartym w pliku, ścieżka dostępu do którego jest wartością opcji. Jeśli zamiast nazwy pliku pojawił się znak `-`, to komenda czyta dane ze standardowego wejścia.
- **-X ścieżka\_dostępu\_do\_pliku** – opcja umożliwia usuwanie wpisów z list kontroli dostępu zgodnie z opisem zawartym w pliku ścieżka dostępu do którego jest wartością opcji.

W pliku, którego nazwa pojawiła się jako wartość opcji **-M** lub **-X** zapisujemy postać wpisu zgodnie z jednym z czterech przedstawionych formatów. Wszystkie białe znaki są pomijane, znak `#` jest znakiem komentarza, który obowiązuje od miejsca jego wystąpienia do znaku przejścia do nowej linii. Dodatkowo w jednej linii może znajdować się jeden wpis. Jeśli w pliku o nazwie *wpis* umieścimy linię w postaci *u:jan:r---*, a następnie komendę *setfacl* wywołamy w następujący sposób:

```
1 [antek@messy Katalog]$ setfacl -M wpis a*
```

to na listach kontroli dostępu wszystkich plików o nazwie rozpoczynającej się na literę *a* pojawi się wpis umożliwiający użytkownikowi *jan* ich odczyt.

### 3.2.3 Właściciel indywidualny i grupowy

W i-węzłach opisujących plik w systemie plików przechowywane są numery identyfikacyjne właściciela indywidualnego i grupowego. Komendy wypisujące informacje o pliku zaczerpnięte z i-węzła domyślnie odnajdują nazwę właściciela i ją wypisują. Numer identyfikacyjny właściciela indywidualnego lub grupowego może pojawić się jako wynik działania odpowiedniej opcji (przykładowo **-n** w komendzie *ls*) lub na skutek niespójności plików konfiguracyjnych użytkowników lub grup (np. w wyniku usunięcia definicji użytkownika z pliku */etc/passwd* lub z pliku */etc/group*).

Właściciel indywidualny oraz grupowy każdego pliku jest określany w momencie jego tworzenia. Zazwyczaj właścicielem indywidualnym staje się użytkownik, który jest właścicielem efektywnym procesu, który wywołał funkcję systemową *create* tworzącą dany plik. Właścicielem grupowym zostaje grupa będąca grupowym, efektywnym właścicielem procesu tworzącego plik. Zazwyczaj jest to grupa podstawowa właściciela indywidualnego chyba, że została zmieniona komendą *sg*. Zarówno indywidualny, jak i grupowy właściciel pliku może zostać zmieniony. Obowiązują dwie zasady:

1. Właściciela indywidualnego może zmienić jedynie użytkownik *root*.
2. Właściciela grupowego może zmienić użytkownik *root* oraz użytkownik będący jego właścicielem indywidualnym pod warunkiem, że należy on do grupy użytkowników, która będzie nowym właścicielem grupowym.

Zmianę właściciela indywidualnego i/lub grupowego przeprowadza się wykorzystując komendę *chown*. Komenda *chgrp* umożliwia jedynie zmianę właściciela grupowego.

Komenda *chown* wymaga podania nazwy nowego właściciela indywidualnego lub grupowego lub obu oraz nazw plików, których właściciele mają zostać zmienieni jako argumentu wywołania. W niektórych implementacjach zamiast jawnie podawać nazwy nowych właścicieli można podać nazwę pliku, którego właściciele będą stanowili wzór. Nazwy nowego właściciela indywidualnego i grupowego oddziela się znakiem `:` lub `..`. Jeśli zmieniamy jednego z właścicieli, nazwę drugiego pomijamy. Stąd, zależnie od potrzeb wykorzystujemy jeden z trzech następujących zapisów:

1. Zmiana obu właścicieli: *nazwa\_użytkownika:nazwa\_grupy* lub *nazwa\_użytkownika.nazwa\_grupy*.
2. Zmiana jedynie właściciela indywidualnego: *nazwa\_użytkownika:* lub *nazwa\_użytkownika.*
3. Zmiana jedynie właściciela grupowego: *:nazwa\_grupy* lub *.nazwa\_grupy*.

Na sposób działania komendy wpływ mają opcje. W dystrybucji RedHat do najczęściej wykorzystywanych zaliczamy:

- **-c** – opcja ustawia tryb gadatliwy pracy komendy, który raportuje jednak tylko te zmiany, które zostały efektywnie przeprowadzone.
- **-h** – jeśli argumentem komendy *chown* jest plik będący dowiązaniem symbolicznym, to zmiana właścicieli obejmuje ten plik, a nie plik na który on pokazuje. Działanie odwrotne możemy wymusić używając przełącznika *—dereference*
- **-form=obecny\_użytkownik:obecna\_grupa** – użycie opcji powoduje, że zmiana właścicieli obejmuje jedynie te pliki, których dotychczasowym właścicielem indywidualnym jest użytkownik o nazwie *obecny\_użytkownik*, zaś grupowym grupa użytkowników o nazwie *obecna\_grupa*.
- **-f** – opcja powoduje, że komenda będzie wypisywała jedynie komunikaty o błędach.
- **-reference=ścieżka\_dostępu\_do\_pliku** – opcja umożliwia zmianę właściciela indywidualnego i grupowego na takich jakich posiada plik, ścieżka dostępu do którego pojawiła się jako wartość opcji.
- **-R** – umożliwia zmianę właścicieli plików w sposób rekursywny, od katalogu wskazanego jako argument wywołania komendy do końca drzewa katalogów.
- **-v** – przełącza raportowanie pracy w tryb gadatliwy, dostarczający informacji o każdej podjętej próbie zmiany właściciela.
- **-H** – jeśli jako argumentu użyto dowiązania symbolicznego wskazującego na katalog, to zmień jego właścicieli.
- **-L** – opcja działa efektywnie z opcją **-R** i powoduje zmianę właścicieli każdego pliku w katalogu, na który pokazuje napotkane w drzewie katalogów dowiązanie symboliczne.
- **-P** – opcja działa efektywnie z opcją **-R** i powoduje zmianę właścicieli jedynie katalogu, na który pokazuje napotkane w drzewie katalogów dowiązanie symboliczne.

Najczęściej wykorzystywane opcje polecenia *chown* w dystrybucji FreeBSD to:

- **-f** – użycie opcji powoduje wstrzymanie wypisywania komunikatów o błędach, które miały miejsce podczas próby zmiany właściciela.

- **-h** – jeśli argumentem wywołania komendy jest dowiązanie symboliczne, to użycie opcji spowoduje, iż zmiana właściciela indywidualnego i/lub grupowego dotyczyć będzie tego pliku, a nie pliku na który dowiązanie pokazuje.
- **-v** – opcja przełącza sposób raportowania w tryb gadatliwy.
- **-R** – użycie opcji umożliwia zmianę właściciela indywidualnego i/lub grupowego w sposób rekursywny, od katalogu będącego argumentem wywołania komendy do końca drzewa katalogów.
- **-H** – efektywne działanie wymaga użycia dodatkowo opcji **-R** i umożliwia zmianę właściciela indywidualnego i/lub grupowego każdego pliku znajdującego się w katalogu, którego nazwa jest argumentem wywołania komendy.
- **-L** – użycie opcji powoduje zmianę właściciela indywidualnego i/lub grupowego we wszystkich katalogach, na które wskazują napotkane w drzewie katalogów dowiązania symboliczne. Dla efektywnego działania opcja wymaga użycia opcji **-R**.
- **-P** – użycie wraz z opcją **-R** powoduje, że plikom znajdującym się w katalogach, na które pokazują napotkane w drzewie katalogów dowiązania nie będą miały zmienionego właściciela indywidualnego i/lub grupowego.

Zmianę tylko właściciela grupowego umożliwia, prostsza składniowo, komenda *chgrp*. Wymaga ona podania jako argumentu nazwy nowego właściciela grupowego oraz nazw plików, których właściciel grupowy ma zostać zmieniony. Opcje komendy *chgrp* są identyczne z opcjami omówionej komendy *chown*. W dystrybucji RedHat kod wykonywalny każdej z nich zapisany jest w innym pliku. Natomiast w dystrybucji FreeBSD kod wykonywalny obu komend zapisany jest w pliku, który widoczny jest pod dwoma nazwami, czyli wykorzystano mechanizm dowiązania twardego. Świadczy o tym chociażby ten sam numer i-węzła opisującego pliki, co przedstawiono poniżej:

```

1 -bash-4.0$ ls -il 'which chown'
2 706863 -r-xr-xr-x  2 root  wheel  6892 Feb 24   2008 /usr/sbin/chown
3 -bash-4.0$ ls -il 'which chgrp'
4 706863 -r-xr-xr-x  2 root  wheel  6892 Feb 24   2008 /usr/bin/chgrp

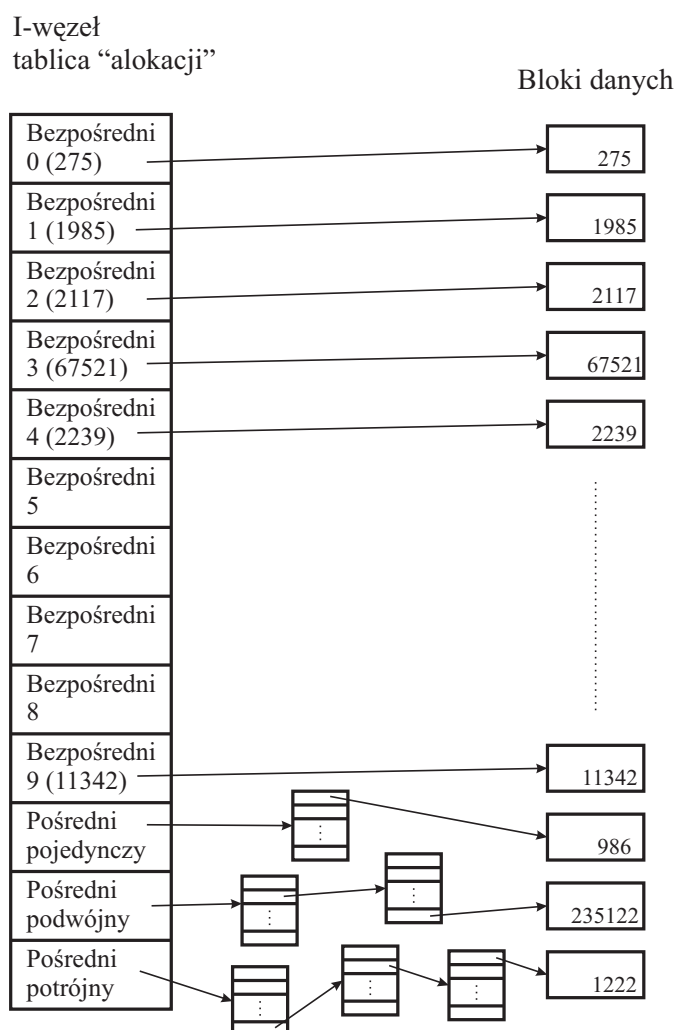
```

### 3.2.4 Rozmieszczenie pliku w systemie plików

I-węzeł będący strukturą opisującą plik zawiera również informację o tym, w których blokach danych systemu plików znajduje się jego zawartość. Przypomnijmy, iż każdy blok danych należący do systemu plików jest rozróżniany numerem, który jednocześnie określa jego pozycję na liście bloków danych. Dla uproszczenia rozważań zakładamy, że blok danych może zawierać dane należące do jednego pliku. Przeciwnostawne wymagania dotyczące małego rozmiaru i-węzła z jednej, a możliwości przechowywania dużych plików z drugiej strony spowodowały, iż w Unix System V pojawiło się rozwiązanie przedstawione na rysunku 3.6.

Lewa strona rysunku 3.6 przedstawia tablicę, która jest przechowywana w i-węźle. W klasycznym rozwiązaniu tablica posiada trzynaście pozycji. Dziesięć pierwszych pozycji to adresy bezpośrednie. Pozycje te zawierają numery bloków danych, w których przechowywane są dane należące do pliku opisywanego przez i-węzeł. Zakładając, iż rozmiar bloku danych wynosi 1 kB, wykorzystując jedynie tę część tablicy, w systemie plików moglibyśmy przechowywać pliki o rozmiarze nie przekraczającym 10 kB. Za to czas dostępu do zapisanych w nich danych jest





Rysunek 3.6: Przykład struktury danych umożliwiającej przechowywanie informacji o rozmieszczeniu danych zapisanych w pliku w blokach danych systemu plików. Alokowane są pojedyncze bloki danych. Różnice w ich numerach wskazują na położenie w różnych obszarach dysku, co negatywnie wpływa na szybkość operacji wejścia/wyjścia wykonywanych na plikach. Rozwiązanie to zastosowano w Unix System V.

relatywnie krótki. Wynika to z faktu, iż proces pracujący na pliku odwołuje się do jego zawartości podając odległość od początku pliku do interesującego go bajtu. Znalezienie bloku danych zawierającego odpowiedni fragment danych wymaga odczytania z dysku przez jądro systemu operacyjnego i-węzła pliku (jeśli jest to pierwszy dostęp do pliku) oraz przeczytania ze znajdującej się w nim tablicy numeru odpowiedniego bloku danych. Znajac numer bloku danych jądro czyta go z dysku, udostępniając zapisane w nim dane procesowi. Jak widać liczba operacji wejścia/wyjścia jest niewielka, a numer odpowiedniego bloku danych odczytywany jest bezpośrednio.

Jak łatwo obliczyć, wykorzystując dziesięć adresów bezpośrednich jesteśmy w stanie, dysponując blokami danych o rozmiarze 1 kB, zapisywać pliki o rozmiarach do 10 kB. Pozycję jedenastą

w tablicy zajmuje adres pośredni pojedynczo. Zawiera on numer bloku danych, w którym zapisana została tablica zawierająca 256 adresów bezpośrednich, a więc numerów bloków danych przechowujących dane. Rozmiar pliku, którego dane jesteśmy w stanie przechować w systemie plików wzrósł do 266 kB ( $(10 + 256) * 1kB$ ). Idąc dalej, pozycja dwunasta zawiera adres pośredni podwójnie. Jest to numer bloku danych, w którym zapisany został numer bloku danych przechowujący tablice 256 elementów, z których każdy to numer bloku danych zawierającego tablicę 256 elementów będących numerami bloków danych przechowujących dane zapisane w pliku. Indeksowanie podwójne znacznie zwiększa możliwości adresowe. Maksymalny rozmiar pliku, który może zostać przechowany wynosi 65802kB ( $(10 + 256 + 256^2) * 1kB$ ), czyli nieco ponad 64 MB. Do dyspozycji pozostaje adres pośredni potrójnie. Jego idea jest identyczna, jak w przypadku adresu pośredniego podwójnie, lecz posiada on o jeden stopień pośredniości więcej. Dzięki temu maksymalny rozmiar pliku, który w systemie plików może być przechowywany wzrasta do 16843018 kB ( $(10 + 256 + 256^2 + 256^3) * 1kB$ ) czyli około 16 GB. W systemach plików dedykowanych dla dużych plików wykorzystuje się adresowanie pośrednie cztero-, a nawet pięciokrotne. Należy jednak zwrócić uwagę, iż wprowadzenie kolejnego stopnia pośredniości pociąga za sobą konieczność wykonania dodatkowej operacji odczytu bloku zawierającego adresy pośrednie oraz operacji przeszukania tablicy adresów w celu znalezienia numeru bloku danych przechowującego odpowiedni fragment danych zapisanych w pliku.

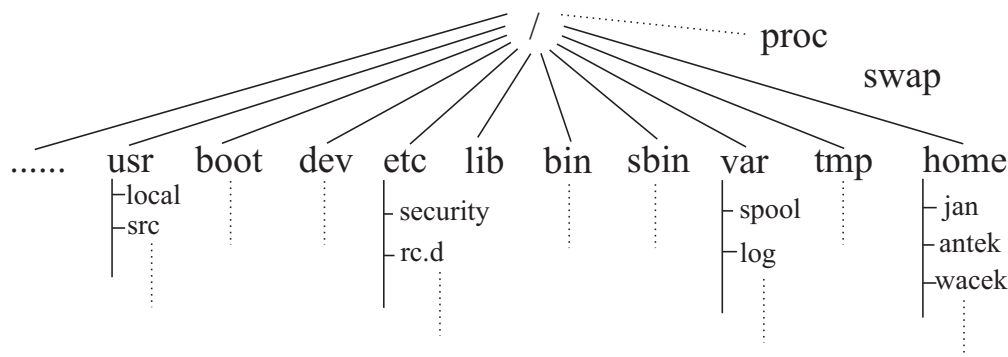
Przedstawiony mechanizm faktycznie umożliwia zachowanie niewielkiego rozmiaru i-węzła, przechowując informacje o rozmieszczeniu danych z pliku w blokach danych systemu plików. Wpływ na powstanie takiego rozwiązania miały również wnioski wyciągnięte z obserwacji liczby i rozmiarów plików przechowywanych w systemie plików. W chwili powstawania opisywanego rozwiązania dominowały pliki o rozmiarze do 10 kB. Stąd użycie dziesięciu adresów bezpośrednich, zapewniających dużą wydajność operacji wykonywanych na pliku. Rosnące rozmiary plików wymusiły powstanie nowych rozwiązań, które mają charakter indywidualny dla danego systemu plików.

### 3.3 Podstawy administracji systemami plików

Plik jest podstawową jednostką służącą do przechowywania informacji w systemie komputerowym. Z punktu widzenia systemu operacyjnego każdy plik zwykły jest ciągiem bajtów, nie posiadającym żadnej wewnętrznej struktury. Dla użytkownika sprawa wygląda nieco inaczej, gdyż z łatwością dokona on podziału plików według różnych, istotnych dla niego kryteriów. Na szybkiego możemy podzielić pliki na te związane z danym projektem, pliki zawierające grafikę jak choćby zdjęcia cyfrowe, które z kolei zostaną podzielone tematycznie lub chronologicznie. Możemy również wyszczególnić pliki, które ogólnie nazwiemy konfiguracyjnymi – najczęściej pliki tekstowe zawierające opis sposobu działania aplikacji. Dla zachowania porządku oraz ogólnej wydajności w dostępie do przechowywanej w plikach informacji mogą one zostać zapisane w katalogach o odpowiedniej nazwie, sugerującej jego zawartość. Katalog, pozwalający przechowywać w sobie pliki i inne katalogi, jest jednostką organizacyjną w drzewie plików, która w zupełności pokrywa potrzeby zwykłego użytkownika jak chodzi o systematyzowanie przechowywanej w plikach informacji i optymalizację dostępu do niej.

Administrator systemu oprócz klasyfikacji znajdujących się w nim plików pod kątem ich typu czy przeznaczenia, stosuje inne kryteria. Dzieli pliki ze względu na: istotność dla działania samego systemu operacyjnego, częstość zmian ich zawartości czy dynamikę zmian ich rozmiaru. Możliwości prostego zarządzania plikami dają katalogi, które pozwalają na „tematyczne” grupowanie plików. Fragment drzewa katalogów przedstawiono na rysunku 3.7. Rozpoczyna go katalog główny, oznaczony symbolicznie przez /, zwany również korzeniem drzewa katalogów (*root*). Ilość

i nazwy znajdujących się w nim katalogów zależą od konkretnej dystrybucji. Często w katalogu głównym administrator tworzy własne katalogi, co może wynikać ze specyfiki użytkowania danego systemu. Spróbujmy określić specyfikę plików przechowywanych w kilku katalogach.



Rysunek 3.7: Fragment drzewa katalogów. Rysunek przedstawia jedynie „najbardziej popularne” katalogi występujące w katalogu głównym zawierające pliki o pewnych, wspólnych cechach.

W katalogu `/home` znajdują się konta użytkowników. Ich zawartość zmienia się często, dlatego relatywnie często należy wykonywać kopię zapasową jego zawartości. Ilość zasobów systemu plików konieczna do przechowywania jego zawartości jest praktycznie nieskończona, gdyż użytkownicy zużywają każdą ich ilość. Pojawia się zatem konieczność ich reglamentowania. Dodatkowo nie możemy dopuścić do sytuacji, w której pliki użytkowników zajmą całe dostępne w systemie plików miejsce, gdyż może to zagrazić stabilnej pracy systemu.

Katalog `/var` jest wykorzystywany m.in. przez system do zapisywania w nim dzienników, przechowywania poczty użytkowników, zadań do drukowania czy zadań do uruchamiania. Ilość zasobów koniecznych do przechowywania zapisywanej w nim informacji jest możliwa do oszacowania już na etapie instalowania systemu operacyjnego. W trakcie jego pracy, nad ilością miejsca zajmowanego przez pliki w katalogu `/var` jesteśmy w stanie panować. Po pierwsze użytkownicy nie mają prawa zapisu w tym katalogu. Dodatkowo podsystemy dziennikowania same dokonują kompresji i rotacji swoich plików, co zostanie dokładnie omówione w rozdziale 4. Stąd reglamentacja zasobów dla zawartości tego katalogu nie jest konieczna.

W katalogu `/usr` przechowywane są istotne fragmenty systemu operacyjnego, jak chociażby biblioteki dołączane dynamicznie, czy moduły jądra systemu operacyjnego. Ilość zasobów koniecznych do przechowywania jego zawartości można oszacować w trakcie instalacji systemu, biorąc pod uwagę oprogramowanie dostępne w danej dystrybucji, które zdecydujemy się zainstalować oraz oprogramowanie dodatkowe wynikające z przyszłych funkcji systemu. Zawartość katalogu zmienia się rzadko, najczęściej wówczas, gdy doinstalowujemy nowe oprogramowanie, co narzuca również częstotliwość wykonywania kopii zapasowej. Oczywiście użytkownicy systemu nie mają prawa tworzenia swoich plików w tym katalogu.

Pliki konfiguracyjne systemu przechowywane są w katalogu `/etc`. Ilość zasobów systemu plików wymagana do przechowywania jego zawartości jest relatywnie niewielka. Zawartość zmienia się jedynie w przypadku zmiany konfiguracji aplikacji lub podsystemu, stąd częstotliwość wykonywania kopii zapasowej nie jest duża. Użytkownicy systemu nie mają prawa zapisywania w nim swoich plików. Nie ma zatem konieczności reglamentowania zasobów systemu plików koniecznych dla przechowywania zawartości katalogu `/etc`.

Jądro systemu operacyjnego oraz inne jego fragmenty konieczne do uruchomienia przechowywane są w katalogu `/boot`. Liczbę zasobów wykorzystywanych przez zawarte w nim pliki można

określić podczas instalowania systemu. Zazwyczaj szacuje się je z zapasem dla ewentualnych kolejnych wersji jądra systemu. Zawartość katalogu praktycznie nie zmienia się. Kopii zapasowych praktycznie nie wykonuje się. Nie ma potrzeby reglamentowania zasobów użytkownikom, gdyż nie mają oni możliwości przechowywania swoich plików w tym katalogu. Należy jednak pamiętać, iż w niektórych dystrybucjach, jak np. we FreeBSD, konieczne jest utworzenie tego systemu plików, gdyż bez niego nie będzie możliwa inicjalizacja systemu operacyjnego.

Katalog */tmp* jest wykorzystywany do chwilowego przechowywania plików przez każdego użytkownika systemu. Specjalne prawa dostępu gwarantują, że plik z katalogu */tmp* może usunąć jedynie jego właściciel i użytkownik *root*. Pliki znajdujące się w katalogu */tmp* są tworzone bezpośrednio przez użytkowników lub też przez uruchomione przez nich aplikacje. Stąd o ilości zasobów koniecznych do przechowywania zapisanych w nim informacji decydują możliwości sprzętowe systemu oraz wymagania zainstalowanych aplikacji. Kopii zapasowej zawartości katalogu praktycznie nie wykonuje się. Nad zawartością katalogu czuwają specjalne procesy, które usuwają pliki znajdujące się w nim dłużej niż określony przedział czasu. Reglamentowanie zasobów użytkownikom jest sprawą indywidualną.

Katalog */proc* jest wirtualnym systemem plików tworzonym przez system. Nie zajmuje on zasobów dyskowych. Jego zawartość stanowią pliki, najczęściej tekstowe, niosące informacje o bieżącym stanie systemu.

*Swap* jest specjalnym systemem plików umożliwiającym działanie mechanizmu pamięci wirtualnej. Wykorzystywana jest przez proces wymiany do przechowywania fragmentów lub całych procesów, których obecność w pamięci o dostępie swobodnym jest niecelowa. Rozmiar określa się podczas instalowania systemu. Może zostać zmieniony w każdym momencie pracy systemu. Zależy on od rozmiaru pamięci o dostępie swobodnym oraz specyfiki uruchamianych w systemie aplikacji.

Śledząc krótką charakterystykę wybranych katalogów łatwo zauważyć specyfikę każdego z nich. Jest ona zależna od rodzaju instalacji. Przykładowo, jeśli system ma pracować na naszym laptopie, czy komputerze domowym, a korzystać z niego będą jedna lub dwie osoby, to wprowadzanie reglamentacji na zasoby systemu plików jest bezsensowne. Po pierwsze sami będziemy kontrolowali ich zajętość. Po drugie aby ograniczenia móc egzekwować musimy wiedzieć ile zasobów zajmują pliki każdego użytkownika, a to wymaga wykorzystania pewnej mocy obliczeniowej. Po trzecie, sami wiemy, które pliki są dla nas istotne i możemy wybiórczo wykonywać ich kopie zapasowe. W systemach przeznaczonych dla szerokiej rzeszy użytkowników sytuacja jest inna. Odseparowanie plików użytkowników od plików systemowych jest konieczne. Aby wykorzystanie zasobów systemu plików było sprawiedliwe konieczne jest zaprowadzenie mechanizmów ich reglamentowania. Administrator musi wykonywać kopię zawartości całego katalogu domowego wszystkich użytkowników, gdyż realizacja indywidualnych potrzeb każdego z nich w tym zakresie przekraczałaby możliwości techniczne.

Katalog nie dostarcza mechanizmów pozwalających chociażby na kontrolowanie ilości zasobów wykorzystywanych przez znajdujące się w nim pliki użytkownika. Nie stanowi również efektywnej granicy pozwalającej na „odseparowanie” plików użytkownika od plików systemowych. Dlatego też stosuje się rozwiązanie, w którym zawartość wybranych katalogów wykorzystuje zasoby oddzielnych systemów plików. Innymi słowy, drzewo katalogów przedstawione na rysunku 3.7 faktycznie nie musi zostać zrealizowane w oparciu o zasoby jednego systemu plików. W praktyce pod drzewem katalogów ukrytych jest kilka systemów plików, których zasoby wykorzystywane są przez różne katalogi. Jak zatem dokonać podziału drzewa katalogów pomiędzy różne systemy plików?

### 3.3.1 Podział drzewa katalogów pomiędzy systemy plików

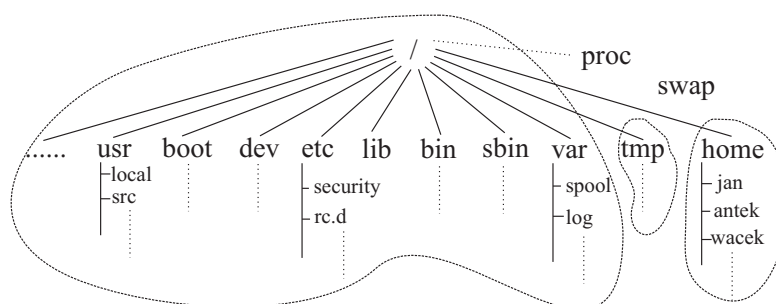
Podziału drzewa katalogów na systemy plików dokonuje się już podczas instalowania systemu. Na tym etapie należy być świadomym, do jakich celów i w jaki sposób będzie wykorzystywany instalowany system. W wyniku procesu instalacji powinniśmy otrzymać system, którego konfiguracja będzie najbliższa tej docelowej, spełniającej nasze wymagania. Oczywiście w pracującym systemie możemy dokonać dowolnych zmian konfiguracyjnych. Spowodują one jednak, że system będzie w pewnych momentach czasu niedostępny, co w przypadku systemów świadczących np. usługi sieciowe natychmiast przełoży się na pieniądze. Właściwy podział drzewa katalogów ma zatem wpływ na bezpieczeństwo systemu oraz na komfort administrowania systemem.

Poniżej przedstawiono dwa przykłady dla skrajnych przypadków podziału drzewa katalogów na komputerze osobistym oraz dla systemu systemu wykorzystywanego przez wielu użytkowników, jako serwer pocztowy jak i system na którym będą oni wykonywali obliczenia wykorzystujące zarówno moc obliczeniową procesora (procesorów) jak i pamięć operacyjną.

#### Drzewo katalogów a systemy plików dla systemu pracującego na komputerze osobistym

Najprostszy podział drzewa katalogów na systemy plików w systemie komputerowym wykorzystywanym przez jednego–dwóch użytkowników może polegać na utworzeniu jednego systemu plików obejmującego całe drzewo katalogów oraz wydzieleniu systemu plików *swap* dla realizacji pamięci wirtualnej.

Rozwiązanie to nie spełnia praktycznie żadnych założeń bezpieczeństwa. Może się bowiem zdarzyć, że jedyny korzystający z systemu użytkownik nawet nieświadomie wykorzysta wszystkie dostępne zasoby systemu plików w postaci bloków danych (np. przez ściągane z internetu pliki multimedialne). Jak wspomniano, może to zakłócić normalną pracę systemu, gdyż część usług i programów systemowych tworzy podczas swej normalnej pracy pliki tymczasowe lub dopisuje do utworzonych plików nowe informacje (przykładowo podsystem dziennikowania).



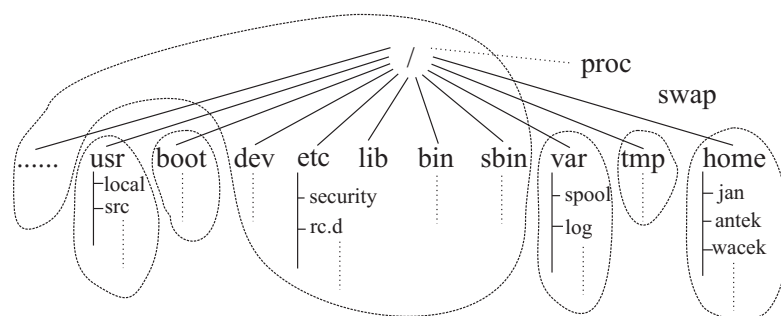
Rysunek 3.8: Przykład podziału drzewa katalogów na systemy plików dla systemów pracujących na komputerach osobistych.

Przedstawiony na rysunku 3.8 przykład podziału drzewa katalogów na systemy plików zapewnia odseparowanie plików użytkownika oraz plików tymczasowych tworzonych w katalogu */tmp* od plików systemowych. W takim rozwiązaniu, jeśli użytkownik wykorzysta zasoby systemu plików, w którym znajduje się jego katalog domowy lub zasoby systemu plików przeznaczonego dla katalogu plików tymczasowych, to nie będzie to miało wpływu na zasoby systemu plików wykorzystywanego przez katalogi systemowe.

### Drzewo katalogów a systemy plików dla serwera przeznaczonego dla wielu użytkowników

Zupełnie inaczej wygląda sytuacja w przypadku systemu udostępniającego zasoby szerokiej rzeszy użytkowników. W takim systemie utworzenie systemu plików dla katalogów domowych jest obowiązkowe. Po pierwsze wpłynie na podniesienie poziomu bezpieczeństwa. Ułatwi również prace administratorskie w zakresie reglamentowania zasobów użytkownikom czy tworzenia kopii zapasowych ich katalogów domowych. Umożliwi również proste, w razie konieczności, zwiększenie rozmiaru systemu plików.

Podobnie sytuacja wygląda w przypadku systemu plików przeznaczonego dla katalogu */tmp*. Wyczerpanie bezpośrednio przez użytkowników lub ich aplikacje jego zasobów w żaden sposób nie wpłynie na pracę systemu operacyjnego. Zaś w przypadku zainstalowania aplikacji, która posiada wymagania w stosunku do zasobów systemu plików */tmp* przekraczającego jego dotychczasowy rozmiar, powiększenie go nie będzie zadaniem trudnym.



Rysunek 3.9: Przykład podziału drzewa katalogów na systemy plików dla systemów komputerowych przeznaczonych dla wielu użytkowników, udostępniających usługi sieciowe.

W proponowanym rozwiązaniu oddzielny system plików przeznaczono również dla katalogu */var*. Na wykorzystanie jego zasobów wpływ ma m.in. intensywność pracy użytkowników. Generują oni zdarzenia, które zapisywane są w plikach dziennika powodując wzrost ich rozmiaru. Zwiększenie liczby użytkowników systemu może spowodować, że zasoby tego systemu plików zostaną wyczerpane. Zastosowane rozwiązanie pozwoli zwiększyć jego rozmiar bez konieczności zmiany rozmiaru innych systemów plików.

Również dla katalogu */boot* utworzony został oddzielny system plików. W systemach komputerowych omawianego przeznaczenia administratorzy często modyfikują jądro systemu lub instalują jego poprawione lub nowsze wersje. Rozwiązanie to pozwala kontrolować zajmowane przez nie zasoby systemu plików. Tworząc oddzielny system plików dla katalogu */boot* należy sprawdzić w dokumentacji systemu ile powinien wynosić jego minimalny rozmiar. Większość dystrybucji wprowadza ograniczenie, które jest egzekwowane już przez program instalatora.

System plików założony dla katalogu */usr* zapewnia izolację oraz efektywną administrację jego zasobami dla istotnych plików systemu operacyjnego.

Dla katalogu głównego */* oraz pozostałych katalogów utworzono oddzielny system plików. Takie rozwiązanie może zostać podyktowane przez proces inicjalizacji systemu operacyjnego, który wymaga aby katalogi */etc*, */lib*, */bin* oraz */sbin* znajdowały się w systemie plików, w którym znajduje się katalog główny */*.

Oczywiście podział drzewa katalogów na systemy plików przeprowadzony podczas instalowania systemu nie musi obowiązywać przez cały czas pracy systemu. W każdej chwili, jeśli zasoby

sprzętowe na to pozwalają, w ramach przedstawionych ograniczeń mogą być tworzone nowe lub łączone istniejące systemy plików.

### 3.3.2 Montowanie systemu plików

#### Podstawy teoretyczne

Dyski fizyczny zawiera zazwyczaj co najmniej jeden wolumen logiczny, który w pracującym systemie jest udostępniany przez program sterujący dysku. Każdy wolumen ma nazwę pliku reprezentującego go w systemie. Procesy sięgają do danych zapisanych w wolumenie otwierając reprezentujący go plik, a następnie czytają zawartą w nim i zapisują do niego informację. Wolumen dyskowy może zawierać system plików, który jak wiemy składa się ogólnie z bloku systemowego, bloku identyfikacyjnego, listy i-węzłów i bloków danych. Jeden i-węzeł jest i-węzłem głównym (zwanym również i-węzłem korzenia) systemu plików, który udostępnia strukturę katalogów systemu plików po wykonaniu funkcji systemowej *mount*.

Funkcja systemowa *mount* dołącza system plików z podanego wolumenu do istniejącej hierarchii w określonym katalogu. Pozwala ona zatem sięgać do danych znajdujących się w wolumenie dyskowym jak do systemu plików, a nie jak do ciągu bloków dyskowych. Warunkiem poprawnego odczytania przez jądro zapisanej w systemie plików informacji jest znajomość jego formatu (wewnętrznej budowy). W praktyce oznacza to tyle, że musi ono posiadać kod umożliwiający korzystanie z danego typu systemu plików. Jak przekonamy się wkrótce, może on zostać skompilowany i stanowić część kodu wykonywalnego jądra lub stanowić oddzielny moduł, który w odpowiednim momencie zostanie przez jądro wykorzystany.

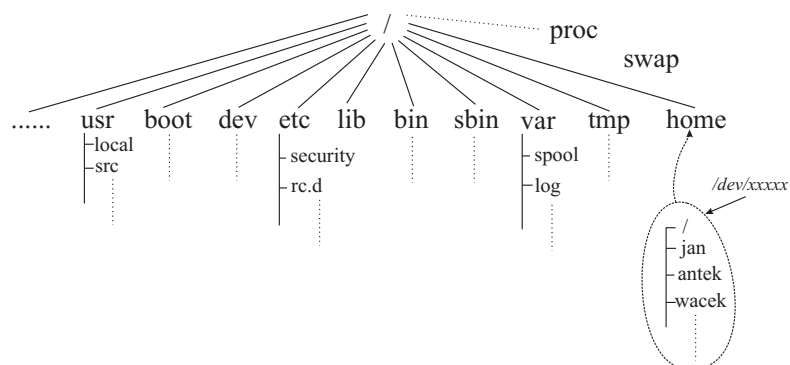
Wywołanie funkcji systemowej *mount* wymaga podania trzech argumentów:

1. Ścieżki dostępu do pliku reprezentującego montowany wolumen dyskowy, zawierający system plików.
2. Ścieżki dostępu do katalogu w istniejącym drzewie katalogów, w którym system plików zostanie zamontowany (tzw. punktu montowania).
3. Opcji montowania umożliwiających wyspecyfikowanie sposobu dostępu do danych przechowywanych w systemie plików (np. tylko do odczytu) lub wpływających na efektywność dostępu.

Jądro systemu operacyjnego przechowuje tzw. *tablicę montowania* (ang. *mount table*) zawierającą struktury opisujące każdy zamontowany system plików. Każda pozycja tablicy zawiera:

- Numer urządzenia, który identyfikuje zamontowany system plików (numer systemu plików).
- Wskaźnik do bufora zawierającego blok identyfikacyjny systemu plików.
- Wskaźnik do i-węzła opisującego korzeń zamontowanego systemu plików.
- Wskaźnik do i-węzła opisującego katalog będący punktem montowania systemu plików.

Algorytm realizowany przez funkcję systemową *mount* składa się z kilku kroków. Jądro znajduje i-węzeł pliku reprezentującego urządzenie na którym znajduje się montowany system plików, pobiera numer urządzenia, który identyfikuje odpowiedni wolumen dyskowy oraz znajduje i-węzeł katalogu, w którym system plików ma zostać zamontowany. Następnie jądro przydziela wolną pozycję w tablicy montowania, zaznacza że jest ona zajęta i nadaje odpowiednią wartość polu przechowującemu numer urządzenia. Oznaczenie danej pozycji w tablicy montowania



Rysunek 3.10: Przykład montowania systemu plików zawierającego katalogi domowe użytkowników w katalogu `/home`. Montowany system plików został założony na urządzeniu, które jest reprezentowane umownie przez plik `/dev/xxxxx`.

jako zajętej zapobiega jej wykorzystaniu w kolejnych wywołaniach funkcji *mount*, zaś zapamiętanie numeru urządzenia umożliwia wychwycenie sytuacji, w której inne procesy próbowałyby zamontować ten sam system plików.

W kolejnym kroku jądro wywołuje funkcję *open* dla urządzenia blokowego, na którym znajduje się system plików. Sprawdzana jest poprawność urządzenia, niekiedy inicjalizowane są struktury danych programu sterującego urządzeniem (sterownika) i wysyłane polecenia inicjalizacji dla sprzętu. Jądro przydziela wolny bufor z puli buforów na blok identyfikacyjny montowanego systemu plików i wczytuje ten blok z urządzenia blokowego. Zapamiętywany jest również wskaźnik do i-węzła katalogu w istniejącym drzewie katalogów, w którym montowany jest system plików, co umożliwia przechodzenie przez punkty montowania podczas analizy ścieżek dostępu zawierających symboliczną nazwę katalogu nadrzędnego, czyli ... Następnie znajdowany jest i-węzeł opisujący katalog główny montowanego systemu plików. W tablicy montowania zapamiętany zostaje wskaźnik do tego i-węzła. W trakcie normalnej pracy systemu plików katalog, w którym system plików został zamontowany (katalog montowania) oraz katalog główny montowanego systemu plików są logicznie równoważne, co zostaje zaimplementowane przez przechowywanie wskaźników do obu opisujących je i-węzłów w jednym rekordzie tablicy montowania. Operacja ta spowodowała, że procesy nie mogą sięgać do i-węzła katalogu montowania, co w praktyce objawia się tym, że jego zawartość jest niedostępna. Zwróćmy uwagę, że nie została przeprowadzona żadna operacja, która spowodowałaby zniszczenie zawartości. Stąd stanie się ona dostępna po wykonaniu operacji odmontowania systemu plików.

Kolejne czynności wykonywane są na strukturach montowanego systemu plików. Jądro inicjalizuje pola bloku identyfikacyjnego, zerując pola blokady listy wolnych bloków danych i i-węzłów oraz ustawiając liczbę wolnych i-węzłów w bloku na zero. Działania te mają na celu ograniczenie niebezpieczeństwa zniszczenia systemu plików podczas jego montowania po awarii systemu, czyli w sytuacji w której nie wykonano odmontowania systemu plików. W wyniku ich wykonania informacja zawarta w bloku identyfikacyjnym daje jądro systemu wrażenie, że w montowanym systemie plików nie ma wolnych i-węzłów. Jądro wykonuje więc algorytm poszukiwania wolnych i-węzłów w systemie plików i uaktualnia ich listę w bloku identyfikacyjnym. Nieaktualna lista wolnych i-węzłów, mogłaby doprowadzić do wykorzystania i-węzła, który faktycznie opisuje istniejący plik prowadząc do utraty zapisanej w nim informacji i powstanie bloków osieroconych. Jeśli natomiast zniszczeniu ulegnie lista wolnych bloków danych, to jądro jej nie naprawi. W takiej sytuacji korzysta się z odpowiednich programów narzędziowych dedykowanych dla konkretnego



typu systemu plików.

Kolejne kroki polegają na uwzględnieniu opcji montowania, co w większości przypadków prowadzi do ustawienia odpowiednich znaczników w bloku identyfikacyjnym. Na zakończenie jądro zaznacza w odpowiedni sposób i-węzeł punktu montowania, aby inne procesy mogły go we właściwy sposób później zidentyfikować.

### Tablica systemów plików

Informacja o systemach plików, na których znajduje się drzewo katalogów dostępne w systemie, przechowywana jest w tzw. tablicy systemów plików. Jest to plik tekstowy przechowywany w katalogu */etc*. Jego nazwa i format zależą od dystrybucji. W większości przypadków plik nosi nazwę *fstab* i posiada budowę linikową, czyli jedna linia opisuje jeden system plików. W niektórych dystrybucjach plik nosi nazwę *filesystems* i posiada budowę zwrotkową. Zwrotkę opisującą system plików rozpoczyna linia z nazwą urządzenia, na którym ten system plików się znajduje, a kolejne linie zawierają wartości opisujących go atrybutów. Informacja stanowiąca zawartość pliku jest wykorzystywana podczas inicjalizacji systemu operacyjnego do zbudowania drzewa katalogów oraz w niektórych przypadkach uruchomienia z linii poleceń komend *mount* oraz *umount*. Stąd, jeśli w systemie utworzymy system plików i chcemy, aby system zarządzał nim tak, jak systemami utworzonymi podczas instalacji systemu, konieczne jest dodanie jego definicji do tablicy systemu plików. Zawartość pliku może modyfikować jedynie użytkownik *root*, natomiast prawo odczytu mają wszyscy użytkownicy zdefiniowani w systemie.

Zobaczmy jak wygląda format tablicy systemu plików. Zawartość pliku */etc/fstab* pochodzącego z dystrybucji RedHat przedstawiono poniżej:

1	UUID=16ccc68d-d792-42e5-9f7c-6fab88a6d4c5	/	ext3	defaults	1	1
2	/dev/sda5	/home	ext3	defaults	1	2
3	LABEL=/tmp	/tmp	ext3	defaults	1	2
4	UUID=40ddceb5-976f-4adc-8815-813be2882fc5	/usr	ext3	defaults	1	2
5	UUID=4830c168-72d4-4205-9597-db34153e66e5	/boot	ext3	defaults	1	2
6	tmpfs	/dev/shm	tmpfs	defaults	0	0
7	devpts	/dev/pts	devpts	gid=5,mode=620	0	0
8	sysfs	/sys	sysfs	defaults	0	0
9	proc	/proc	proc	defaults	0	0
10	UUID=1abb4020-53b3-4a5c-bf85-915d3a5e39f9	swap	swap	defaults	0	0

Jak wspomniano jedna linia opisuje jeden system plików. Każda linia jest podzielona na sześć pól. Separatorem jest biały znak, co najmniej jeden. Kolejne pola oznaczają:

1. Pole pierwsze opisuje urządzenie blokowe, na którym znajduje się montowany system plików. Opis ten może mieć kilka postaci. W przypadku lokalnych systemów plików może to być bezwzględna ścieżka dostępu do pliku urządzenia<sup>3</sup>.

Ten sposób definiowania pojawił się w linii pliku */etc/fstab*. Wadą tego rozwiązania jest to, że zmiana konfiguracji sprzętowej może prowadzić do sytuacji, w której zmienia się nazwy urządzeń i dotychczasowa tablica systemów plików przestanie być aktualna.

Urządzenie blokowe może zostać również rozpoznane po etykiecie, która zostaje mu nadana podczas tworzenia. Ten sposób opisu został przedstawiony w liniach 3, 4 oraz 10. Składa

<sup>3</sup>Nazwy plików urządzeń w dystrybucji RedHat zostały omówione w podrozdziale 3.4, zaś w dystrybucjach FreeBSD w podrozdziale 3.5.

się on z napisu *LABEL=* oraz etykiety. Ten sposób opisu posiada pewną wadę, która może ujawnić się po zainstalowaniu w systemie dysku pochodzącego z innego systemu, a wykorzystującego ten sam schemat etykietowania. Wówczas prawdopodobieństwo, że w systemie pojawią się dwa systemy plików o identycznej etykiecie jest duże, a jeśli tak się stanie, to pojawią się niejednoznaczność i w efekcie żaden z systemów nie zostanie zamontowany. Etykietę systemu plików można w dystrybucji RedHat zmienić korzystając z komendy *e2label*.

Wyliminowanie wad dotychczas omówionych sposobów opisu urządzenia ma zapewnić mu nadanie unikalnego numeru identyfikacyjnego urządzenia (ang. *Universally Unique Identifier*). Opis w tablicy systemu plików składa się z napisu *UUID=* oraz występującego za nim identyfikatora unikalnego. Gdyby jednak w systemie pojawiły się dwa urządzenia o identycznym identyfikatorze, lub jeśli zmiana identyfikatora ułatwi nam konfigurowanie systemu, to w dystrybucji RedHat można skorzystać z komendy *tune2fs*. Przykładowo, wartości atrybutów urządzenia blokowego uzyskamy uruchamiając komendę *tune2fs* z opcją **-l** podając jako argument ścieżkę dostępu do pliku urządzenia:

```

1 [root@messy ~]# tune2fs -l /dev/sda5
2 tune2fs 1.40.8 (13-Mar-2008)
3 Filesystem volume name:   /home
4 Last mounted on:          <not available>
5 Filesystem UUID:          1081a212-91f4-4193-9538-1c2e99ad3efa
6 .....

```

Chcąc zmienić wartość identyfikatora musimy użyć opcji **-U** podając jako jej wartość postać nowego identyfikatora oraz jako argument wywołania ścieżkę dostępu do pliku urządzenia.

W przypadku montowania sieciowych systemów plików, opis urządzenia ma format *host:katalog*, gdzie *host* oznacza adres IP lub adres symboliczny hosta, zaś *katalog* ścieżkę dostępu do udostępnionego na hoście katalogu.

2. Drugie pole zawiera ścieżkę dostępu do katalogu, w którym system plików ma zostać zamontowany. W skrócie jest on nazywany punktem montowania.
3. Pole trzecie opisuje typ systemu plików. W systemach linuksowych najczęściej wykorzystywanymi są: *ext2*, *ext3*, *ext4*, *iso9660*, *proc*, *swap*, *msdos*, *vfat*, *xfs*, *raiserfs*, *jfs*, *sysfs*. Pole to stanowi informacje dla jądra systemu. To czy obsłuży ono system plików faktycznie znajdujący się na urządzeniu, w szczególności czy zostanie on zamontowany, zależy od tego czy jądro posiada funkcje obsługujące jego format.
4. W czwartym polu zostały zawarte opcje montowania w postaci listy identyfikatorów opcji oddzielonych przecinkiem. Zawierają one informacje o tym, w jaki sposób dany system może być dostępny oraz inne ustawienia wpływające głównie na jego efektywność. W opcjach montowania może pojawić się także informacja o sposobie reglamentowania jego zasobów. Opis podstawowych opcji znajduje się w następnym punkcie.
5. Z informacji zapisanej w polu piątym korzysta komenda *dump*. Jeśli wartość zapisanej w nim liczby wynosi 0, to program *dump* nie będzie wykonywał kopii zapasowej danego systemu plików. Wartość różna od 0 określa minimalny poziom zrzutu, przy którym dojdzie do skopiowania tego systemu plików.

6. Ostatnie, szóste pole przechowuje informację dla komendy *fsck*. Zapisane w nim wartości oznaczają kolejność, w jakiej ma zostać zakończony proces sprawdzania spójności systemu plików przed ich zamontowaniem. Wartość 0 oznacza, że dany system nie musi podlegać weryfikacji. Główny system plików powinien posiadać dla tego argumentu wartość 1 co oznacza, że proces jego weryfikacji powinien zostać zakończony przed pozostałymi systemami plików. Dla systemów plików, których spójność przed zamontowaniem ma zostać sprawdzona, powinna zostać nadana wartość 2. Jeśli wartość pola dla dwóch lub większej liczby systemów plików jest identyczna, to o kolejności decyduje kolejność, w jakiej zostały one wyspecyfikowane w pliku.

Tablicę systemów plików pochodzącą z systemu FreeBSD przedstawiono poniżej:

#	Device	Mountpoint	FStype	Options	Dump	Pass#
1	/dev/ad0s1b	none	swap	sw	0	0
2	/dev/ad0s1a	/	ufs	rw	1	1
3	/dev/ad0s1e	/home	ufs	rw,acfs	2	2
4	/dev/ad0s1f	/tmp	ufs	rw	2	2
5	/dev/ad0s1d	/usr	ufs	rw	2	2
6	/dev/ad0s1g	/var	ufs	rw	2	2
7	/dev/acd0	/cdrom	cd9660	ro,noauto	0	0

Również w tym systemie posiada ona budowę linijkową, a każda linijka składa się z sześciu pól. Pierwszą różnicą w stosunku do tablicy z dystrybucji RedHat jest występowanie komentarzy. Tradycyjnie są to komentarze linijkowe. Rozpoczynają się znakiem *#*, a kończą znakiem przejścia do nowej linii. Ich występowanie pozwala szybko zorientować się w znaczeniu kolejnych pól osobie, która sięga do pliku po raz pierwszy lub po długiej przerwie. Znaczenie kolejnych pól linii opisujących montowane systemy plików w systemie FreeBSD jest bardzo podobne do omówionego w systemie RedHat. Pola oznaczają:

- Pole oznaczone nagłówkiem *Device* zawiera nazwę urządzenia, na którym znajduje się dany system plików. W przypadku lokalnych systemów plików jest to ścieżka dostępu do pliku urządzenia<sup>4</sup>. Jeśli zamontowany ma zostać sieciowy system plików, to jest on zdefiniowany napisem w formacie *host:katalog*, gdzie *host* oznacza adres IP lub adres symboliczny hosta, zaś *katalog* ścieżkę dostępu do udostępnionego na hoście katalogu.
- Nagłówek drugiego pola, *Mountpoint*, wskazuje jednoznacznie, że zawiera ono ścieżkę dostępu do katalogu, w którym dany system plików ma zostać zamontowany. W przypadku pliku wymiany pojawia się napis *none*.
- Pole trzecie, oznaczone jako *FStype*, przechowuje informacje o typie systemu plików, który ma zostać zamontowany. W przypadku systemu FreeBSD lista obsługiwanych typów jest nieco krótsza i obejmuje: *ufs* – uniksowy system plików Fast File System w obu wersjach, *msdos* – system plików typu FAT, *mfs* – Memory File System, *swap* – format systemu plików dla pliku wymiany oraz *cd9660* – CD-ROM. Możliwe jest także wykorzystanie linuksowych systemów plików ext2, xfs i Reiser4 oraz rozpowszechnianego na licencji CDDL systemu plików firmy Sun o nazwie ZFS (*Zettabyte File System*).
- Kolejne pole, z nagłówkiem *Options* zawiera opcje montowania, mówiące w jaki sposób jądro ma traktować dany system plików. Zostaną one dokładnie opisane w następnym punkcie.

<sup>4</sup>Systematykę nazwy plików urządzeń w dystrybucji FreeBSD omówiono w podrozdziale 3.5.

- W polu *Dump* zawarta jest informacja dla komendy *dump*. Identycznie, jak w dystrybucji RedHat jest to liczba mówiąca czy należy wykonywać kopię danego systemu plików. Wartość 0 oznacza, że program *dump* nie będzie jej wykonywał, zaś każda większa od zera oznacza poziom zrzutu, przy którym dojdzie do skopiowania danego systemu plików.
- Ostatnie pole oznaczone jako *Pass#* określa w której fazie procesu rozruchowego należy zamontować dany system plików. Wartość 0 oznacza, że dany system nie będzie montowany podczas inicjalizacji systemu. Tylko główny system plików jest oznaczony wartością 1, co oznacza, że ma on zostać zamontowany jako pierwszy. Pozostałe systemy plików zostały oznaczone wartością 2, co oznacza, że mają zostać zamontowane po głównym systemie plików.

### Komenda *mount*

Komenda *mount* służy do dołączania (montowania) systemu plików do drzewa katalogów. Dostarcza również informacji o systemach plików znajdujących się w systemie. W niektórych dystrybucjach po zamontowaniu systemu plików komenda *mount* uaktualnia informację w pliku o nazwie *mtab* znajdującym się w katalogu */etc*. Jego budowa jest zbliżona do budowy pliku */etc/fstab*. Prawa dostępu pozwalają na odczyt jego zawartości każdemu użytkownikowi systemu. Modyfikację może przeprowadzić jedynie użytkownik *root*.

Zacznijmy od omówienia takich jej uruchomień, które nie dokonują montowania. W dystrybucji RedHat mają one następującą postać:

- Uruchomienie bez opcji i argumentów powoduje wypisanie informacji o systemach plików (fizycznych i wirtualnych) znajdujących się w systemie. Ma ona następującą postać:

```

1 20:40 [bory@thorin ~]$ mount
2 /dev/hda2 on / type ext3 (rw)
3 none on /proc type proc (rw)
4 none on /sys type sysfs (rw)
5 none on /dev/pts type devpts (rw,gid=5,mode=620)
6 usbfs on /proc/bus/usb type usbfs (rw)
7 none on /dev/shm type tmpfs (rw)
8 /dev/hda3 on /home type ext3 (rw)
9 /dev/hda6 on /tmp type ext3 (rw)
10 none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
11 sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
```

Jedna linia opisuje jeden system plików. Każdą rozpoczyna informacja o pliku urządzenia, na którym dany system plików został założony. Wyjątkiem są wirtualne systemy plików, oznaczone słowem *none*, lub specjalne systemy plików wykorzystywane np. przez mechanizm zdalnego wywołania procedury, oznaczone jako *sunrpc* czy uniwersalny system plików dla urządzeń USB, oznaczone jako *usbfs*. Następnie pojawia się informacja o punkcie montowania oraz typie zamontowanego systemu plików. Linie zamykają opcje montowania, określające sposób dostępu do danego systemu plików.

- Opcja **-l** pozwala nieznacznie zmodyfikować format wypisanej informacji, a mianowicie linie opisujące rzeczywiste systemy plików są zakończone zapisaną w nawiasach kwadratowych ścieżką dostępu do punktu montowania.

- Opcja **-t**, po której pojawia się typ systemu pełni w tym przypadku rolę filtra, powodując iż wypisana informacji dotyczy jedynie systemów plików podanego typu. Wynik użycia komendy *mount* z opcją **-l** oraz **-t** ograniczającą dostarczaną informację do systemów plików typu *ext3* przedstawiono poniżej:

```
1 21:52 [bory@thorin ~]$ mount -l -t ext3
2 /dev/hda2 on / type ext3 (rw) [/]
3 /dev/hda3 on /home type ext3 (rw) [/home]
4 /dev/hda6 on /tmp type ext3 (rw) [/tmp]
```

Jak widać, w powyższy sposób komendę *mount* może uruchomić każdy użytkownik systemu. Pozostałe sposoby uruchomienia, których wynikiem ma być zamontowanie systemu plików są efektywnie dostępna dla użytkownika *root*. Wyjątkiem są systemy plików wyspecyfikowane w tablicy systemów plików, które w opcjach montowania mają wyspecyfikowane numery identyfikacyjne lub nazwę użytkowników upoważnionych do montowania i odmontowania.

Chcąc dokonać montowania systemów plików zdefiniowanych w tablicy, należy uruchomić komendę *mount* z opcją **-a**. W tym wypadku możliwe jest użycie dodatkowych opcji służących sprecyzowaniu, które z systemów plików zdefiniowanych w tablicy mają zostać zamontowane. Do dyspozycji mamy dwie podstawowe opcje:

1. **-t** – znana już opcja, pozwalająca w tym przypadku na zamontowanie systemów plików, których typy zostały wymienione jako jej wartość.
2. **-O** – pozwala na zamontowanie systemów plików, których opcje montowania w tablicy systemów plików są identyczne z tymi, które zostały wyspecyfikowane po opcji. Przykładowo komenda:

```
1 mount -a -O no_netdev
```

spowoduje zamontowanie wszystkich systemów plików zdefiniowanych w tablicy, które w opcjach montowania mają wyspecyfikowaną opcję **no\_netdev**. Należy zwrócić uwagę na fakt, że napis **no** jest po opcji **-O** traktowany jako fragment nazwy opcji, a nie jak negacja, co ma miejsce w przypadku użycia opcji **-t**.

W przypadku użycia obu opcji, komenda *mount* dokonuje zamontowania systemów plików, spełniających oba kryteria. Przykładowo, w wyniku wykonania poniższej komendy:

```
1 mount -a -t xfs -O _netdev
```

zostaną zamontowane systemy plików zdefiniowane w tablicy, które są typu *xfs* oraz w ich opcjach montowania pojawił się napis **\_netdev**.

Może jednak okazać się, że chcemy zamontować jeden system plików. Zaczniemy od przypadku, w którym system ten jest zdefiniowany w tablicy systemów plików. Wówczas komendę *mount* wystarczy uruchomić podając jako argument punkt montowania lub ścieżkę dostępu do pliku urządzenia. Reszta informacji zostanie zaczerpnięta z tablicy systemów plików. Należą do nich również opcje montowania, których listę można rozszerzyć korzystając z opcji **-o**.

Ostatnia możliwość dotyczy montowania systemu plików, który nie został zdefiniowany w tablicy systemów plików. W takim przypadku komenda *mount* wymaga podania dwóch argumentów. Pierwszym jest ścieżka dostępu do pliku urządzenia, gdyż musimy powiedzieć *co* montujemy. Jak łatwo się domyśleć, drugim argumentem jest punkt montowania, czyli ścieżka dostępu

do katalogu *gdzie* system plików zostanie zamontowany. Tak uruchomiona komenda `mount` użyje domyślnych opcji montowania, których listę możemy rozszerzyć stosując opcję `-o`. Będzie również starała się rozpoznać typ montowanego systemu plików, który możemy podpowiedzieć wartością opcji `-t`. Przykładowo:

```
1 [root@messy ~]# mount -t xfs -o acl /dev/sda7 /scratch4
```

W wyniku wykonania powyższej komendy w katalogu `/scratch4` zostanie zamontowany system plików typu `xfs` znajdujący się na urządzeniu `/dev/sda7`. Oprócz domyślnych opcji montowania, użyto opcję `acl`, która umożliwi kontrolowanie praw dostępu do znajdujących się w nim plików z wykorzystaniem list kontroli dostępu.

Przedstawione sposoby uruchomienia komendy `mount` wykorzystywały jedynie podstawowe jej opcje. Uzupełnijmy informacje o nich oraz poznajmy kilka nowych, do których należą:

- **-i** – dostępne w systemie różne typy systemów plików posiadają własne programy montujące, które zazwyczaj znajdują się w katalogu `/sbin`, a ich nazwa odpowiada schematowi `mount.<identyfikator_typu>`. Użycie opcji spowoduje, że nie zostaną one uruchomione, nawet jeśli istnieją, a montowania dokona program `/bin/mount`.
- **-n** – opcja pozwala na zamontowanie systemu plików bez uaktualniania informacji zapisanej w pliku `/etc/mtab`. Jest wykorzystywana jeśli system plików, w którym przechowywana jest zawartość katalogu `/etc` nie jest zamontowany.
- **-r** – komenda uruchomiona z tą opcją montuje system plików w trybie tylko do odczytu.
- **-w** – montowanie w trybie do odczytu i zapisu. Opcja ta jest domyślna.
- **-L etykieta** – opcja umożliwia zamontowanie systemu plików, którego etykieta jest zgodna z podaną jako wartość opcji.
- **-U identyfikator** – komenda użyta z tą opcją dokona montowania systemu plików, którego identyfikator jest identyczny z podanym jako wartość opcji.
- **-t identyfikator\_typu** – opcja służy podaniu typu systemu plików, który ma zostać zamontowany. Na chwilę obecną lista identyfikatorów typów jest następująca: `adfs`, `affs`, `autofs`, `coda`, `coherent`, `cramfs`, `devpts`, `efs`, `ext`, `ext2`, `ext3`, `ext3`, `hfs`, `hpfs`, `iso9660`, `jfs`, `minix`, `msdos`, `ncpfs`, `nfs`, `nfs4`, `ntfs`, `proc`, `qnx4`, `ramfs`, `reiserfs`, `romfs`, `smbfs`, `sysv`, `tmpfs`, `udf`, `ufs`, `umsdos`, `usbfs`, `vfat`, `xenix`, `xfs`, `xiafs`. Montowania większości z nich dokonuje komenda `mount` z katalogu `/bin`. Może ona uruchomić program montujący dany typ systemu plików, jeśli ten istnieje (jest to program z pliku `/sbin/mount.<identyfikator_typu>`). Jeśli jako inedyfikatora typu użyjemy napisu `auto`, to komenda `mount` spróbuje rozpoznać typ montowanego systemu plików na podstawie zawartości jego bloku systemowego. Nie dla wszystkich wymienionych typów jest to jednak możliwe.
- **-o** – opcja ta umożliwia określenie sposobu obsługi przez jądro montowanego systemu plików. Opcje montowania specyfikuje się po opcji **-o** jako listę, w której separatorem jest przecinek. Opcje montowania możemy podzielić na ogólne, obsługiwane zwykłym programem montującym (`/bin/mount`) oraz specyficzne dla danego typu systemu plików, dostępne w programach `mount` dedykowanych właśnie temu typowi. Do najczęściej wykorzystywanych, ogólnych opcji montowania należą:
  - **async** – wszystkie operacje zapisu i odczytu wykonywane na danych i metadanych montowanego systemu plików będą wykonywane jako asynchroniczne.

- **atime** – przechowywany w i-węzle czas dostępu do pliku będzie uaktualniany przy każdym dostępie. Opcja domyślna.
- **auto** – umożliwia montowanie danego systemu plików przez komendę *mount* użytą z opcją **-a**. Opcja domyślna.
- **defaults** – oznacza montowanie danego systemu plików z opcjami domyślnymi. Są nimi opcje: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, **async**.
- **dev** – jeśli w montowanym systemie plików znajdują się pliki urządzeń blokowych lub znakowych, to jądro systemu ma je traktować jako pliki urządzeń.
- **exec** – opcja umożliwia uruchamianie plików wykonywalnych z montowanego systemu plików.
- **\_netdev** – system plików znajduje się na urządzeniu dostępnym przez sieć. Opcja ta zapewnia, że dany system plików nie zostanie zamontowany zanim nie będzie dostępny interfejs sieciowy.
- **noatime** – opcja zapewnia, że w i-węzłach danego systemu plików nie będzie aktualizowany czas dostępu do pliku. Opcja jest wykorzystywana do zwiększania wydajności systemu plików.
- **noauto** – system plików oznaczony tą opcją może być montowany bezpośrednio, tzn. że nie zostanie zamontowany przez komendę *mount* wywołaną z opcją **-a**.
- **nodev** – jeśli w montowanym systemie plików znajdują się pliki urządzeń blokowych lub znakowych, to jądro systemu ma je traktować jak pliki zwykłe.
- **noexec** – opcja uniemożliwia uruchamianie programów zapisanych w plikach znajdujących się w danym systemie plików. Jest ona wykorzystywana np. dla systemów plików znajdujących się na serwerach, a przechowujących programy dla innych architektur procesorowych.
- **nosuid** – użycie opcji powoduje, że w stosunku do plików znajdujących się w danym systemie plików nie działają prawa SUID oraz SGID.
- **nouser** – efektywnie opcja powoduje, że dany system plików może zamontować jedynie użytkownik *root*. Opcja ta jest domyślna.
- **remount** – opcja pozwala na przemontowanie zamontowanego systemu plików. Praktycznie wykorzystywana dla zmiany opcji montowania, np. jeśli chcemy uczynić zamontowany system plików dostępny w trybie do odczytu.
- **ro** – użycie opcji spowoduje, iż dany system plików zostanie zamontowany w trybie do odczytu (bez możliwości zmiany zawartości jakiegokolwiek znajdującego się w nim pliku).
- **rw** – opcja umożliwia montowanie danego systemu plików w trybie do zapisu i odczytu.
- **suid** – po zamontowaniu systemu plików, jądro będzie interpretowało prawa dostępu SUID oraz SGID do znajdujących się w nim plików.
- **sync** – operacje wejścia/wyjścia na danych i metadanych systemu plików zamontowanego z tą opcją będą wykonywane w sposób synchroniczny.
- **dirsync** – użycie opcji spowoduje, że w danym systemie plików operacje zmiany zawartości katalogów będą wykonywane w sposób synchroniczny. Dotyczy to następujących funkcji systemowych: *creat*, *link*, *unlink*, *symlink*, *mkdir*, *rmdir*, *mknod* oraz *rename*.

- **user** – umożliwia zwykłemu użytkownikowi, którego nazwa pojawiła się po opcji, montowanie danego systemu plików. Nazwa użytkownika zostaje zapisana w pliku */etc/mtab*, co powoduje, że może on również odmontować dany system plików. Użycie tej opcji powoduje, że zostają także użyte opcje: **noexec**, **nosuid** oraz **nodev**. Stąd sekwencja opcji: **user,noexec,nosuid,nodev** daje efekt działania jedynie opcji **user**.
- **users** – montowanie systemu plików wykorzystującego tę opcję może przeprowadzić każdy użytkownik systemu. Użycie tej opcji automatycznie wymusza użycie opcji: **noexec**, **nosuid** oraz **nodev**. Podobnie jak w przypadku opcji **user** sekwencja opcji: **user,noexec,nosuid,nodev** daje efekt działania jedynie opcji **users**.
- **–bind** – opcja komendy *mount* dostępna w systemach linuksowych, umożliwiająca zamontowanie w innym katalogu aktualnie zamontowanego systemu plików. System plików staje się dostępny z wielu punktów montowania. Dostępny od wersji 2.5 jądra systemu operacyjnego.
- **–move** – opcja umożliwiająca zmianę punktu montowania zamontowanego systemu plików (przeniesienie zamontowanego systemu plików w drzewie katalogów). Dostępny również w systemach linuksowych z jądrem w wersji co najmniej 2.5.

Opcje charakterystyczne dla wybranych typów systemów plików dostępnych w systemach linuksowych zostaną omówione w następnym podrozdziale traktującym o budowie i administracji wybranych typów systemów plików.

W systemie FreeBSD komenda *mount* uruchomiona bez opcji i argumentów wypisuje listę zamontowanych w systemie systemów plików, której postać jest następująca:

```

1 -bash-4.0$ mount
2 /dev/ad0s1a on / (ufs, local)
3 devfs on /dev (devfs, local)
4 /dev/ad0s1e on /home (ufs, local, soft-updates, acls)
5 /dev/ad0s1f on /tmp (ufs, local, soft-updates)
6 /dev/ad0s1d on /usr (ufs, local, soft-updates)
7 /dev/ad0s1g on /var (ufs, local, soft-updates)

```

Każda linia opisuje jeden zamontowany system plików. Informacja w niej zapisana składa się ze ścieżki dostępu do pliku urządzenia oraz punktu montowania. Trzeci fragment, zapisany w nawiasach, zawiera informacje o: typie zamontowanego systemu plików, czy jest to system plików znajdujący się na urządzeniach lokalnego systemu czy dostępny na innym systemie np. poprzez sieć. Informację zamykają opcje montowania. Zmianę formatu do występującego w tablicy systemów plików zapenia opcja **-p**:

```

1 -bash-4.0$ mount -p
2 /dev/ad0s1a      /                ufs      rw          1 1
3 devfs           /dev            devfs    rw          0 0
4 /dev/ad0s1e     /home           ufs      acls        2 2
5 /dev/ad0s1f     /tmp            ufs      rw          2 2
6 /dev/ad0s1d     /usr            ufs      rw          2 2
7 /dev/ad0s1g     /var            ufs      rw          2 2

```



Użycie komendy *mount*, wynikiem działania której jest zamontowanie systemu plików jest możliwe przez użytkownika *root*. Od tej reguły istnieją nieliczne wyjątki, które zostaną przedstawione przy okazji omawiania opcji montowania. Najprostsze uruchomienie polega na użyciu opcji **-a**. Wówczas zamontowane zostaną wszystkie systemy plików opisane w tablicy systemów plików, za wyjątkiem już zamontowanych oraz tych, których opcje montowania zawierają opcje *noauto* lub *late*. Z opcją **-a** w dystrybucji FreeBSD związane są następujące opcje:

- **-F** *ścieżka\_dostępu\_do\_tablicy\_systemów\_plików* – umożliwia podanie ścieżki dostępu do pliku zawierającego tablicę systemów plików, innego niż */etc/fstab*.
- **-l** – umożliwia zamontowanie systemu plików nawet jeśli w tablicy systemu plików w opcjach jego montowania występuje opcja *late*.
- **-t** – opcja ta umożliwia wyspecyfikowanie typu montowanego systemu plików. Użycie jej z opcją **-a** spowoduje, że zamontowane zostaną jedynie te systemy plików, które zostały wyspecyfikowane w tablicy systemów plików, które nie są zamontowane i których typ jest identyczny z podanym jako wartość opcji. Możliwe jest użycie negacji. Przykładowo komenda:

```
1 -bash-4.0# mount -a -t noufs
```

spowoduje zamontowanie wszystkich systemów plików wyspecyfikowanych w tablicy za wyjątkiem już zamontowanych oraz tych, których typ został określony jako *ufs*.

Najczęściej komendę *mount* wykorzystuje się do montowania jednego, konkretnego systemu plików. Ten sposób montowania może dotyczyć zarówno systemów plików, które zostały zdefiniowane w tablicy systemów plików, jak również tych, których definicja nie została jeszcze w niej zapisana. W pierwszym przypadku, czyli dla zamontowania systemu plików zdefiniowanego w tablicy, w wywołaniu komendy *mount* wystarczy podać, jako argument nazwę pliku urządzenia lub punkt montowania. Brakująca informacja zostanie pobrana z tablicy systemów plików. Montowanie systemu plików, którego definicja nie znajduje się w tablicy systemu plików wymaga, w najprostszym przypadku podania dwóch argumentów: pierwszym jest ścieżka dostępu do pliku urządzenia, na którym znajduje się dany system plików, drugim zaś ścieżka dostępu do katalogu, w którym dany system plików ma zostać zamontowany. Dodatkowo można użyć opcji **-t** umożliwiającej podanie typu montowanego systemu plików (może być to *ufs* lub inny, obsługiwany przez jądro systemu operacyjnego) oraz opcji **-o** pozwalającej na określenie opcji montowania.

Oczywiście, w każdym przypadku uruchomienia komendy *mount*, można użyć dostępnych w niej opcji. Do najpopularniejszych w systemie FreeBSD należą:

- **-d** – opcja umożliwia śledzenie wykonania procesu montowania. Wykorzystywana z opcją **-v**.
- **-f** – może zdarzyć się, że zmiana trybu pracy z zapis/odczyt na tylko do zapisu zamontowanego systemu plików z pewnych powodów nie jest możliwa. Opcja ta wymusza przejście stąd uwaga o rozsądnym jej stosowaniu.
- **-o** – opcja umożliwia podanie opcji montowania danego systemu plików w postaci listy, w której separatorem jest przecinek. Jeśli wyspecyfikowane w liście opcje są sprzeczne, to efektywnie działa opcja położona na liście bardziej na prawo. Do najczęściej wykorzystywanych opcji komendy *mount* w systemie FreeBSD należą:

- **acls** – umożliwia wykorzystanie w montowanym systemie plików list kontroli dostępu, zarządzanych komendami *setfacl*, *getfacl* oraz *chacl*.
- **async** – wszystkie operacje wejścia/wyjścia na danym systemie plików będą wykonywane asynchronicznie. Użycie tej opcji nie jest zalecane, gdyż może prowadzić do utraty spójności przechowywanej informacji. Stąd warunkiem użycia jest posiadanie odpowiednich programów narzędziowych.
- **current** – stosowana z opcją **-u** umożliwia montowanie już zamontowanego systemu plików z tymi samymi opcjami.
- **force** – działanie identyczne do opcji **-f**, a więc umożliwia „siłowe” zamontowanie już zamontowanego systemu plików z ograniczeniem dostępu, przykładowo zamontowanie systemu plików w trybie tylko do odczytu podczas, gdy do momentu wywołania komendy był on zamontowany do zapisu i odczytu.
- **fstab** – opcja umożliwia zamontowanie systemu plików z opcjami zapisanymi w tablicy systemów plików. Stosowana z opcją **-u**.
- **late** – stosowana z opcją **-a** powoduje, że systemy plików, które zostały wyspecyfikowane w tablicy plików z tą opcją nie zostaną zamontowane. Montowanie tych systemów plików zapewnia opcja **-l**.
- **multilabel** – opcja umożliwia wykorzystanie mechanizmu wieloetykietowej kontroli dostępu do plików w nim przechowywanych, jeśli system plików posiada ten mechanizm zaimplementowany.
- **noasync** – użycie opcji powoduje, że operacje wejścia/wyjścia na metadanych będą prowadzone w sposób synchroniczny zaś na danych w sposób asynchroniczny.
- **noatime** – opcja spowoduje, że system nie będzie dokonywał uaktualnienia daty dostępu do pliku przechowywanej w i-węźle, jeśli plik był otwierany w trybie do odczytu. Opcja zwiększa wydajność systemu plików, zwłaszcza jeśli przechowuje on dużą liczbę plików z często modyfikowaną zawartością.
- **noauto** – komenda *mount* uruchomiona z opcją **-a** nie zamontuje systemu plików, w opcjach montowania którego w tablicy systemów plików pojawiła się ta opcja.
- **nocluster** – opcja wyłącza odczyt danych w postaci zgrupowanych bloków danych.
- **noclusterw** – użycie opcji spowoduje, że dane w systemie plików nie będą zapisywane w postaci grup bloków danych znajdujących się na sąsiednich cylindrach dysku.
- **noexec** – użycie opcji uniemożliwia uruchamianie programów zapisanych w plikach binarnych przechowywanych w danym systemie plików. Opcja wykorzystywana w systemach pełniących rolę serwerów przechowujących programy dla różnych architektur komputerowych.
- **nosuid** – opcja powoduje, że w danym systemie plików nie są respektowane rozszerzone prawa dostępu SUID oraz SGID.
- **nosymfollow** – użycie opcji „wyłącza” w danym systemie plików działanie dowiązań symbolicznych. Dowiązania istnieją bez zmian, ale pliki na które wskazują nie są przez nie dostępne.
- **ro** – opcja umożliwia zamontowanie danego systemu plików w trybie tylko do odczytu. Działanie analogiczne do opcji **-r** komendy *mount*.
- **snapshot** – użycie opcji spowoduje wykonanie kopii migawkowej systemu plików. Wymagane jest użycie opcji **-u**. Pliki zawierające kopie są przechowywane wewnątrz

danego systemu plików. Maksymalna ich liczba to 20. Informacja o kopii preferowanej jest przechowywana w bloku systemowym systemu plików. Dokładny opis zarządzania kopiami zapasowymi można znaleźć w manualu komendy *mount*.

- **suid** – katalogi tworzone w systemie plików zamontowanych z tą opcją będą zachowywały się tak, jak gdyby miały ustawione prawo SGID. W praktyce każdy utworzony w katalogu plik regularny lub katalog będzie dziedziczył po nim właściciela grupowego. Dodatkowo plikom regularnym nie będzie nadawane prawo wykonywania. Opcja ta jest często wykorzystywana dla systemów plików, w których przechowywane są katalogi współdzielone z wykorzystaniem protokołu SMB.
- **sync** – wszystkie operacje wejścia/wyjścia na metadanych i danych przechowywanych w danym systemie plików będą wykonywane w sposób synchroniczny.
- **update** – podobnie jak opcja **-u** komendy wskazuje, że opcje montowania danego systemu plików uległy zmianie.
- **-r** – użycie opcji umożliwia zamontowanie danego systemu plików w trybie tylko do odczytu (równoważne użyciu opcji: **-o r**). Ten sposób montowania jest wykorzystywany przykładowo podczas wykonywania kopii zapasowej.
- **-t** – opcja pozwala na wyspecyfikowanie typu montowanego systemu plików. Domyślnie jest to typ *ufs* – system plików będący unowocześnioną wersją systemu FFS. Komenda *mount* uruchomiona z opcją **-t** przekazuje opcje montowania bezpośrednio do komendy systemowej *nmount*. W przypadku, w którym typ montowanego systemu plików jest inny niż domyślny, a jednocześnie jednym z następujących: *cd9660*, *mfs*, *msdosfs*, *nfs*, *nfs4*, *ntfs*, *nuwfs*, *nullfs*, *portals*, *smbfs*, *udf* oraz *unionfs* komenda *mount* usiłuje uruchomić program z katalogu */sbin* o nazwie *mount\_TYP*, gdzie napis *TYP* oznacza jeden z wymienionych identyfikatorów systemów plików. Zatem, podobnie jak w systemach operacyjnych rodziny RedHat, do montowania określonego typu systemu plików wykorzystywane są dedykowane programy umożliwiające wykorzystanie specyficznych opcji montowania.
- **-u** – opcja pozwala na zmianę trybu pracy zamontowanego systemu plików. Jest przykładowo wykorzystywana do zmiany trybu pracy na tylko do odczytu, czy w przypadku wprowadzenia w zamontowanym systemie plików list kontroli dostępu lub reglamentacji zasobów.
- **-v** – komenda uruchomiona z tą opcją dostarcza szczegółowego opisu podejmowanych czynności i ich efektów (tryb gadatliwy).
- **-w** – użycie opcji umożliwia zamontowanie danego systemu plików w trybie do odczytu i zapisu (tryb normalnej pracy systemu plików).

### 3.3.3 Odmontowanie systemu plików

#### Podstawy teoretyczne

Odmontowanie systemu plików jest dokonywane przez funkcję systemową *umount*, której wywołanie wymaga podania argumentu będącego ścieżką dostępu do pliku reprezentującego urządzenie, na którym system ten się znajduje.

Funkcja *umount* realizuje algorytm, który podobnie jak w przypadku montowania systemu plików składa się z kilku kroków. W pierwszej kolejności jądro sięga do i-węzła pliku opisującego odmontowywane urządzenie, odnajduje numer urządzenia, zwalnia i-węzeł i znajduje w tablicy

montowania rekord zawierający ten sam numer urządzenia. Przed efektywnym wykonaniem odmontowania systemu plików, jądro sprawdza, czy żaden plik z tego systemu nie jest używany. Praktycznie polega to próbie znalezienia w tablicy i-węzłów plików z numerem urządzenia takim samym, jak numer demontowanego systemu. Aktywne pliki mają dodatni licznik odwołań i obejmują pliki, które są bieżącymi katalogami procesów, pliki zawierające dzielone instrukcje procesów, które są właśnie wykonywane i otwarte pliki, których nie zamknięto. Jeśli w systemie plików znaleziono co najmniej jeden aktywny plik, to wykonanie funkcji *umount* kończy się błędem. Poszukiwanie aktywnych plików można zrealizować komendą *fuser*.

Bufory dyskowe mogą jednak nadal zawierać bloki przeznaczone do zapisu opóźnionego, których zawartość nie została jeszcze zapisana na dysk, więc jądro opróżnia te bufory. Usuwa także pozycje w tablicy segmentów, które odpowiadają instrukcjom dzielonym i nie są używane. Następnie zapisuje na dysk wszystkie ostatnio zmodyfikowane bloki identyfikacyjne i uaktualnia kopie dyskowe wszystkich i-węzłów, które wymagają aktualizacji. W dalszej kolejności jądro zwalnia i-węzeł katalogu głównego zamontowanego systemu plików i wywołuje program sterujący urządzeniem zawierającego system plików w celu zamknięcia urządzenia. Następnie przegląda bufory w puli buforów i zwalnia te, które zawierają bloki zamkniętego właśnie systemu plików przesuwając je na początek listy wolnych buforów. Zeruje znacznik wskazujący punkt zamontowania w i-węźle i zwalnia i-węzeł. Na zakończenie zwalnia pozycję w tablicy montowania.

Po poprawnym zakończeniu wykonania funkcji *umount* zawartość katalogu, w którym system był zamontowany staje się dostępna przez opisujący go i-węzeł. Informacja przechowywana w odmontowanym systemie plików nie jest niszczone. Nie ma jedynie do niej dostępu, gdyż w systemie brak jest struktury mającej dostęp do i-węzła katalogu głównego systemu plików.

### Komenda *umount*

Odmontowanie systemu plików może wykonać użytkownik *root* lub użytkownik, którego nazwa lub numer identyfikacyjny pojawiły się w opcjach montowania lub zostały wyspecyfikowane w pliku */etc/fstab*. Służy do tego komenda *umount*. Przypomnijmy, że nie można odmontować systemu plików, który zawiera choć jeden plik otwarty przez proces lub w którym choć jeden katalog jest bieżącym dla uruchomionego w systemie procesu. Wykonanie komendy *umount* kończy się wówczas błędem, a wypisywany komunikat jest następującej treści:

```
1 [root@messy home]# umount /home
2 umount: /home: device is busy
```

Komenda *fuser* wypisuje numery identyfikacyjne procesów (PID), które pracują na wskazanych plikach lub wykorzystują zasoby systemów plików. W najprostszym przypadku jej uruchomienia wymagane jest podanie jako argumentu nazwy co najmniej jednego pliku, katalogu, a w przypadku systemu plików punktu montowania lub nazwy pliku reprezentującego urządzenie na którym znajduje się interesujący nas system plików. W przypadku sprawdzania wykorzystania systemu plików konieczne jest użycie opcji *-c*, która dokona sprawdzenia wszystkich plików i katalogów w systemie plików, a nie jedynie we wskazanych jako argumenty wywołania komendy. Użycie dodatkowo opcji *-v* pozwoli uzyskać informacje w postaci długiej.

W przypadku dystrybucji RedHat, użycie komendy *fuser* dla urządzenia */dev/sda5*, na którym znajduje się system plików zamontowany w katalogu */home* dało następujący wynik:

```
1 [root@messy home]# fuser -vc /dev/sda5
2                               USER      PID ACCESS COMMAND
3 /dev/sda5:                   jan        7893 ..c.. bash
```

4	root	7933	..c..	su
5	root	7934	..c..	bash
6	jan	8001	..c..	bash
7	jan	8068	F.ce.	otw_test

Użycie opcji **-v** umożliwiło otrzymanie informacji w postaci tabelki. Druga linia powyższego listingu zawiera jej nagłówek. Pierwsza, nieopisana kolumna przechowuje informacje o pliku lub katalogu, który podlegał sprawdzeniu. Kolumna *USER* zawiera nazwę użytkownika, którego proces wykorzystuje zasoby systemu plików. Kolumna *PID* numer identyfikacyjny procesu. Sposób wykorzystania zasobów został opisany w kolumnie *ACCESS*. Nazwa procesu znajduje się w ostatniej kolumnie nazwanej *COMMAND*.

W dystrybucji RedHat opis sposobu wykorzystania pliku lub zasobów systemu plików składa się z pięciu pól. W każdym polu może pojawić się konkretna litera lub kropka, jeśli ten sposób nie ma miejsca. Rozróżniane sposoby to:

- c – katalog, którego nazwa pojawiła się jako argument wywołania komendy jest katalogiem bieżącym dla danego procesu. Jeśli jako argumentu użyto punktu montowania systemu plików lub nazwy pliku urządzenia, na którym on się znajduje, to proces posiada katalog bieżący w drzewie katalogów tego systemu plików.
- e – proces jest właśnie w stanie wykonywania.
- f – proces posiada otwarty plik, ale komenda nie podaje trybu, w jakim został on otwarty.
- F – proces posiada otwarty plik w trybie do zapisu.
- r – katalog, którego nazwa pojawiła się jako argument wywołania jest katalogiem głównym.
- m – plik jest mapowany do przestrzeni wirtualnej, lub jest to biblioteka współdzielona (ładowana dynamicznie), której fragmenty są aktualnie załadowane przez proces.

Interpretując przykładowe informacje dostarczone przez komendę *fuser* możemy stwierdzić, że w systemie plików znajdującym się na urządzeniu reprezentowanym przez plik */dev/sda5* znajduje się katalog bieżący procesów *bash* użytkownika *jan* (linie 3 i 6), procesu *su* użytkownika *root* (linia 4) oraz procesu *bash* (linia 5). Użytkownik *jan* uruchomił proces o nazwie *otw\_test* (linia 7), który posiada katalog bieżący w rozpatrywanym systemie plików, posiada w nim także otwarty do zapisu plik i jest aktualnie wykonywany.

Dotychczas wykorzystane zostały dwie opcje komendy *fuser*. W dystrybucji RedHat do najczęściej wykorzystywanych należą:

- **-a** – umożliwia wypisanie informacji o wszystkich plikach, a nie jedynie o tych, które są aktualnie wykorzystywane.
- **-k** – opcja ta umożliwia wysłanie sygnału do procesu wykorzystującego plik lub zasoby systemu plików, którego nazwa pojawiła się jako argument wywołania komendy. Wysyłany jest sygnał *SIGKILL*. Jeśli do procesu ma zostać wysłany inny sygnał, to można go wyspecyfikować w formacie *-signal*, gdzie *signal* oznacza nazwę sygnału.
- **-i** – użycie opcji przełączy komendę do pracy w trybie interaktywnym, co spowoduje konieczność potwierdzenia wysłania sygnału do każdego procesu, co pokazano na poniższym listingu:

```

1 [root@messy ~]# fuser -cv /home
2             USER      PID ACCESS COMMAND
3 /home:      jan        8692 .c... bash
4             root       8732 .c... su
5             jan        8798 .c... bash
6             jan        8838 F.ce. otw
7 [root@messy ~]# fuser -kic /home
8 /home:      8692c 8732c 8798c 8838ce
9 Kill process 8692 ? (y/N) n
10 Kill process 8732 ? (y/N) n
11 Kill process 8798 ? (y/N) n
12 Kill process 8838 ? (y/N) y

```

- **-l** – opcja umożliwia wypisanie nazw wszystkich sygnałów.
- **-m nazwa** – wartością opcji może być ścieżka dostępu do pliku znajdującego się w zamontowanym systemie plików lub ścieżka dostępu do pliku urządzenia na którym znajduje się zamontowany system plików. Użycie opcji spowoduje wypisanie informacji o wszystkich procesach korzystających z plików w wyspecyfikowanym katalogu lub systemie plików. Jest to sposób alternatywny do podania jako argumentów nazw katalogów lub systemów plików.
- **-n obszar** – gdzie napis *obszar* może przyjąć jedną z trzech wartości, a to:
  1. **name** – domyślna, oznacza, że przedmiotem naszego zainteresowania są pliki lub systemy plików określane ścieżką dostępu.
  2. **udp** – śledzimy wykorzystanie lokalnych portów UDP. Może zostać podany numer portu lub nazwa. Składnia ma następującą postać ogólną: *nazwa lub numer/nazwa*. Przykładowo: 21/udp.
  3. **tcp** – podobnie jak powyżej, ale dla lokalnych portów TCP.

Postać komendy dla znalezienia w systemie procesów pracujących na lokalnym porcie 22 z wykorzystaniem protokołu TCP jest następująca:

```

1 [root@messy ~]# fuser -v -n tcp 22/tcp
2             USER      PID ACCESS COMMAND
3 22/tcp:     root       1918 F.... sshd
4             jan        8691 F.... sshd
5             root       8792 f.... sshd
6             jan        8797 F.... sshd

```

- **-s** – opcja przełącza komendę do trybu pracy, w którym żadne informacje nie są wypisywane na ekranie. Jeśli jednocześnie użyte zostaną opcje **-u** lub **-v** to zostaną one zignorowane. Opcji tej nie używamy z opcją **-a**.
- **-signal** – umożliwia wyspecyfikowanie sygnału, który ma zostać wysłany do procesu. Wymagane jest podanie nazwy sygnału. Przykładowo pojawienie się w opcjach wywołania komendy napisu *-TERM* spowoduje, że do procesu zostanie wysłany sygnał *terminate* o numerze 15.

- **-u** – użycie opcji powoduje, że w informacjach dostarczanych przez komendę do każdego numeru procesu dołączona zostanie nazwa użytkownika w systemie, który jest jego właścicielem. Przykładowo:

```
1 [root@messy ~]# fuser -cu /home
2 /home:                8692c(jan)  8732c(root)  8798c(jan)  8926ce(jan)
```

- **-v** – umożliwia raportowanie w postaci tabeli z czterema kolumnami. W trybie domyślnym komenda zwraca jedynie numery identyfikacyjne procesów.
- **-4** – opcja wymusza wyszukiwanie soketów protokołu IP jedynie w wersji IV. Nie można jej stosować jednocześnie z opcją **-6**.
- **-6** – opcja wymusza wyszukiwanie soketów protokołu IP jedynie w wersji VI. Nie można jej stosować jednocześnie z opcją **-4**.

Tyle jeśli chodzi o komendę *fuser* dostępną w systemach z rodziny RedHat. W dystrybucji FreeBSD niestety komenda ta nie występuje. Jak zatem znaleźć procesy, które korzystają z plików znajdujących się w systemach plików, które chcemy odmontować? Rozwiązanie problemu stanowi komenda *lsof*. Tu miła rzecz, a mianowicie we wszystkich systemach, w których jest dostępna posiada identyczne opcje oraz format dostarczanej informacji. Niestety nie we wszystkich systemach jest dostępna w instalacji podstawowej. W systemie FreeBSD, podobnie jak w NetBSD i OpenBSD należy ją zainstalować pobierając pakiet z repozytorium. Proces instalacji pakietów oprogramowania w tej dystrybucji zostanie dokładnie omówiony w tomie trzecim. Pakiet *lsof* jest dostępny także dla systemów komercyjnych takich jak np. AIX, Solaris czy HP-UX. W dystrybucji RedHat jest dostępny w podstawowej instalacji.

Pomimo, że składnia komendy jest jednolita, to niestety skomplikowana. Ograniczymy się zatem do przykładów, które dostarczą informacji o procesach korzystających z plików znajdujących się w odmontowywanym systemie plików. Przykład użycia komendy przedstawiono poniżej:

```
1 -su-4.0# lsof +f -- /home
2 lsof: WARNING: compiled for FreeBSD release 7.2-PRERELEASE; this is 7.0-RELEASE.
3 COMMAND  PID  USER  FD   TYPE DEVICE SIZE/OFF  NODE NAME
4 bash     937  antek  cwd   VDIR  0,92      0 23552 /home/antek
5 su       939   root  cwd   VDIR  0,92      0 23552 /home/antek
6 bash     955   jan   cwd   VDIR  0,92      0 70656 /home/jan
7 otw_test 957   jan   cwd   VDIR  0,92      0 70656 /home/jan
8 otw_test 957   jan   3w    VREG  0,92      0 70667 /home/jan/plik.txt
```

Zacznijmy od omówienia składni. Jako argument pojawiła się ścieżka dostępu do katalogu */home*. Komendę uruchomiono z opcją **f**. Przed opcją może pojawić się znak **-** informujący, że argument należy traktować jak katalog. Znaku **+** używamy, jeśli w katalogu będącym argumentem zamontowany został system plików, a komenda ma dostarczyć informacje o wszystkich procesach, które wykorzystują znajdujące się w nim pliki lub katalogi. Występowanie znaków **--** wynika ze składni. Służą one do grupowania opcji i oddzielania ich od argumentów (separator).

Informacja dostarczana przez komendę *lsof* jest zawarta w 9-ciu kolumnach. Zawierają one:

1. Kolumna *COMMAND* nazwę procesu, który korzysta z plików lub katalogów wyspecyfikowanego katalogu lub systemu plików.
2. Kolumna *PID* numer identyfikacyjny procesu.

3. Kolumna *USER* nazwę użytkownika w systemie będącego właścicielem procesu.
4. Kolumna *FD* może zawierać numer pliku (ang. *File Descriptor*) i/lub opis sposobu wykorzystania danego pliku lub katalogu. Spośród najczęściej spotykanych opisów wymienić należy:

- **cwd** – dany katalog jest katalogiem bieżącym procesu.
- **pd** – katalog nadrzędny (ang. *Parent Directory*).
- **rtd** – katalog główny.
- **mem** – plik mapowany w pamięci.
- **txt** – plik programu zawierający kod wykonywalny i dane.

Za numerem deskryptora pliku może pojawić się informacja o trybie otwarcia pliku. Jest ona w postaci jednej litery, a odpowiednie litery oznaczają:

- **r** – plik otwarty do odczytu.
- **w** – plik otwarty do zapisu.
- **u** – plik otwarty do odczytu i zapisu.
- **–** – tryb otwarcia pliku nie jest znany.

Plik może również posiadać założoną przez proces blokadę. Informację o sposobie działania tego mechanizmu dostarczają następujące litery:

- **r** – blokada odczytu fragmentu pliku.
- **R** – blokada odczytu całego pliku.
- **w** – blokada zapisu fragmentu pliku.
- **W** – blokada zapisu całego pliku.
- **u** – blokada zapisu i odczytu dowolnej części pliku (w tym całości).
- **U** – występuje blokada innego typu.

5. Kolumna *TYPE* informuje o typie węzła skojarzonego z danym plikiem. Przykładowo:
  - *VDIR* – katalog.
  - *VREG* – plik regularny.
  - *IPv4* – plik typu socket, wykorzystywany przez protokół IP w wersji IV.
  - *BLK* – plik urządzenia blokowego.
  - *LINK* – dowiązanie symboliczne.
6. Kolumna *DEVICE* zawiera numer urządzenia, na którym znajduje się dany plik lub katalog. Są to liczby *minor* oraz *major* charakteryzujące plik urządzenia.
7. Kolumna *SIZE/OFF* to rozmiar pliku lub numer bajtu od początku pliku od którego rozpocznie się wykonywanie następnej operacji zapisu lub odczytu.
8. Kolumna *NODE* przechowuje numer i-węzła opisującego plik w lokalnym systemie plików.
9. Kolumna *NAME* zawiera bezwzględną ścieżkę dostępu do wykorzystywanego pliku.



Komenda *lsof* nie posiada możliwości wysyłania sygnałów do procesów wykorzystujących zasoby odmontowywanego systemu plików. Znając jednak ich numery identyfikacyjne, możemy w tym celu wykorzystać komendę *kill*, omówioną w następnym rozdziale.

Założmy zatem, że zasoby odmontowywanego systemu plików nie są wykorzystywane przez żaden proces. Możemy zatem wywołać komendę *umount*.

W systemie FreeBSD komenda *umount* wymaga podania jako argumentu ścieżki dostępu do katalogu, w którym zamontowany jest system plików lub ścieżki do pliku urządzenia, na którym znajduje się system plików lub numeru identyfikacyjnego systemu plików. Numer identyfikacyjny zwraca komenda *mount* użyta z opcją **-v**. W takim wywołaniu komendy najczęściej korzysta się z dwóch opcji:

1. **-f** – umożliwia odmontowanie systemu plików nawet jeśli znajdują się w nim pliki lub katalogi wykorzystywane przez procesy. Po odmontowaniu, pierwsza operacja wykonana przez proces na pliku lub katalogu zakończy się błędem. Korzystając z opcji **-f** nie można odmontować głównego systemu plików.
2. **-v** – użycie opcji powoduje wypisanie dodatkowych informacji o odmontowanym systemie plików.

Z użyciem opcji możemy również specyfikować inne sposoby odmontowywania. Takie wywołanie komendy *umount* nie wymaga podania argumentu. Do wykorzystywanych opcji należą:

- **-a** – użycie opcji umożliwia odmontowanie wszystkich systemów plików opisanych w tablicy, czyli pliku */etc/fstab*.
- **-A** – opcja powoduje odmontowanie wszystkich systemów plików, które w momencie jej wywołania były zamontowane.
- **-F ścieżka\_dostępu\_do\_pliku\_fstab** – opcja wymaga podania jako wartości ścieżki dostępu do pliku zawierającego tablicę systemów plików, według zawartości którego będzie przebiegało ich odmontowywanie (plik alternatywny do */etc/fstab*).
- **-h nazwa\_hosta** – opcja ta jest wykorzystywana do odmontowywania sieciowych systemów plików, które fizycznie znajdują się na hoście, którego nazwa symboliczna lub adres IP zostały podane jako wartość opcji.
- **-t lista\_typów** – umożliwia odmontowanie systemu plików opisanego w tablicy systemów plików, którego typ jest zgodny z wyspecyfikowanym w liście będącej wartością opcji. Przykładowo komenda *umount -af -t nfs* spowoduje odmontowanie wszystkich sieciowych systemów plików. Lista zbudowana jest z identyfikatorów typów oddzielonych przecinkami. Negację zapewnia specyfikator **no**, który może wystąpić przed listą. W takiej sytuacji lista zawierać będzie typy systemów plików, które mają zostać nie odmontowane.

### 3.3.4 Reglamentowanie zasobów systemu plików - kontyngenty dyskowe (quota)

Każdy zasób systemu komputerowego jest ograniczony. Jest powszechnie wiadomo, że w systemach wieloużytkownikowych najszybciej wyczerpywalnymi zasobami są: na pierwszym miejscu papier w drukarce sieciowej, a zaraz za nim miejsce na dysku, w systemie plików dostępnym dla użytkowników. Oczywiście dostępność obu zasobów można reglamentować. W niniejszym rozdziale zajmiemy się ograniczaniem zasobów systemów plików, zostawiając sprawy ograniczania zasobów sieciowych do tomu III.

Quota jest funkcjonalnością systemu plików co oznacza, że może ona zostać zdefiniowana dla konkretnego systemu plików. Ponieważ zwykli użytkownicy nie mają prawa zapisu w katalogach przechowujących np. oprogramowanie systemowe, więc wprowadzanie kontyngentów w systemach plików, w których katalogi te się znajdują jest niecelowe. Stąd między innymi pojawiło się szerokie omówienie podziału drzewa katalogów na systemy plików w systemach wieloużytkownikowych oraz uwaga o oddzielaniu plików użytkowników od plików systemowych przez przechowywanie ich w oddzielnych katalogach. Praktycznie kontyngenty dyskowe wprowadza się w systemach plików przechowujących katalogi domowe użytkowników, w katalogu */tmp* i innych wspólnych katalogach przeznaczonych do chwilowego przechowywania dużych plików oraz w katalogach przechowujących pliki pocztowe. Quota może zostać zdefiniowana dla użytkownika lub dla grupy użytkowników. Zasadniczo rozróżnia się dwa rodzaje ograniczeń, które są przyznawane. Są to:

1. **Twarde** – po ustawieniu nie może zostać przekroczone. Próba przekroczenia powoduje, że bieżąca operacja zapisu kończy się błędem.
2. **Miękkie** – jest często przyznawane z limitem czasu (*grace period*). Może zostać przekroczony dowolną ilość razy, ale za każdym razem na czas nie dłuższy niż określony limitem czasu. Przekroczenie limitu czasu powoduje, że nie ma możliwości zapisu nowych informacji w systemie plików mimo, iż ograniczenie twarde nie zostało osiągnięte. Należy wówczas zwolnić wykorzystywane zasoby tak, aby zejść poniżej ograniczenia miękkiego.

W systemie plików limitowaniu podlegają dwa rodzaje zasobów. Są to bloki danych, których dostępna ilość przekłada się na rozmiar przechowywanych plików oraz liczba i-węzłów mówiąca o możliwej do utworzenia liczbie plików.

Reasumując, system quota działa w obrębie systemu plików i może ograniczać liczbę dostępnych bloków danych oraz liczbę i-węzłów, a ograniczenia dotyczą pojedynczego użytkownika lub grupy użytkowników. Nałożone ograniczenia mogą być twarde czyli nieprzekraczalne lub miękkie czyli możliwe do przekroczenia, ale za zwyczaj na pewien okres czasu.

### Quota w systemach rodziny RedHat

Administrowanie systemem kontyngentów składa się z dwóch kroków. Pierwszym jest jego skonfigurowanie, polegające na zamontowaniu właściwych systemów plików z opcjami, które umożliwiają wprowadzenie ograniczeń na zasoby dyskowe dla użytkowników indywidualnych oraz grupy użytkowników. Następnie w katalogu głównym danego systemu plików tworzy się pliki bazy danych przechowujące rekordy z informacją o aktualnie wykorzystanych zasobach przez użytkowników indywidualnych i grupy użytkowników. Ich nazwy zależą od wersji systemu kontyngentów. Natomiast zawartość bazy danych jest uaktualniania w ściśle określonych momentach czasu.

Mając przygotowany system możemy rozpocząć administrowanie nim. Polega ono głównie na dodawaniu do stworzonej bazy danych użytkowników i grupy ograniczeń dla nowodefiniowanych użytkowników i grup, określaniu indywidualnych poziomów ograniczeń i limitu czasu, oraz ew. zatrzymywania i uruchamiania systemu.

**Przykład działania systemu** Przykład zastosowania będzie dotyczył systemów plików */home* oraz */tmp*. Rozpoczynamy od dopisania opcji montowania do pliku */etc/fstab*. Opcja *usrquota* umożliwi limitowanie zasobów użytkownikom indywidualnym, zaś *grpquota* grupom. Linie opisujące wybrane systemy plików mają teraz następującą postać:

1	UUID=47decf0f-2a8d-4b30-bb0a-f5809fcd1958	/home	ext3	defaults,usrquota,grpquota	1	2
2	UUID=efbc2f2a-7dfc-47aa-b3b2-7498086f21e0	/tmp	ext3	defaults,usrquota,grpquota	1	2

Dodanie odpowiednich opcji do pliku */etc/fstab* spowoduje właściwe montowanie systemów plików przy uruchamianiu systemu. Można również dokonać przemontowania systemów plików w działającym systemie operacyjnym:

```
1 [root@messy ~]# mount -o remount /home
2 [root@messy ~]# mount -o remount /tmp
```

Kolejnym krokiem będzie stworzenie plików baz danych. Służy do tego komenda *quotacheck*. W wyniku jej uruchomienia, w katalogach głównych powstają stosowne pliki:

```
1 [root@messy ~]# quotacheck -cug /home
2 [root@messy ~]# quotacheck -cug /tmp
3 [root@messy ~]# ls -l /home/aqu*
4 -rw----- 1 root root 7168 2010-08-26 16:53 /home/aquota.group
5 -rw----- 1 root root 7168 2010-08-26 16:53 /home/aquota.user
6 [root@messy ~]# ls -l /tmp/aqu*
7 -rw----- 1 root root 6144 2010-08-26 16:54 /tmp/aquota.group
8 -rw----- 1 root root 6144 2010-08-26 16:54 /tmp/aquota.user
```

Teraz uruchamiamy system kontyngentów poleceniem *quotaon*. Jeśli chcemy zrobić to dla wybranych systemów plików, to ich punkty montowania pojawiają się jako argumenty wywołania. Uruchomienie systemu kontyngentów dla wszystkich systemów plików, w których ma on działać najprościej osiągnąć stosując opcję *-a*. Stąd mamy:

```
1 [root@messy ~]# quotaon -a
```

Od tego momentu system działa. Jego działanie polega jednak tylko na zliczaniu zasobów wskazanych systemów plików, wykorzystywanych przez użytkowników i grupy. System można wyłączyć poleceniem *quotaoff*, którego składnia jest identyczna jak polecenia *quotaon*. Użytkownik może już sprawdzić narzucone ograniczenia oraz wykorzystanie zasobów. Służy do tego polecenie *quota*, ale należy go użyć z opcją *-v*:

```
1 [antek@messy ~]$ quota -v
2 Disk quotas for user antek (uid 501):
3   Filesystem  blocks    quota   limit   grace   files   quota   limit   grace
4   /dev/sda5   6756      0       0       0       48      0       0
5   /dev/sda3   472       0       0       0       23      0       0
```

Wypisana informacja składa się z nagłówka opisującego użytkownika z podaniem jego nazwy i numeru identyfikacyjnego oraz tabelki. Nagłówek opisuje kolejne kolumny, którymi są:

1. *Filesystem* – nazwa pliku urządzenia reprezentującego dany system plików.
2. *blocks* – liczba bloków dyskowych wykorzystanych przez pliki użytkownika w danym systemie plików.
3. *quota* – ograniczenie miękkie na wykorzystaną liczbę bloków dyskowych. Wartość 0 mówi, że ograniczenie nie jest nałożone.

4. *limit* – ograniczenie twarde na liczbę bloków dyskowych. Wartość 0 informuje o braku ograniczenia.
5. *grace* – wartość limitu czasowego na przekroczenie ograniczenia miękkiego zajmowanych bloków danych.
6. *files* – liczba wykorzystanych i-węzłów.
7. *quota* – ograniczenie miękkie na wykorzystaną liczbę i-węzłów. Wartość 0 oznacza brak ograniczenia.
8. *limit* – ograniczenie twarde na wykorzystaną liczbę i-węzłów. Wartość 0 oznacza brak ograniczenia.
9. *grace* – wartość limitu czasowego na przekroczenie ograniczenia miękkiego zajmowanych i-węzłów.

Jak widać użytkownik nie ma narzuconych żadnych ograniczeń. Kolejnym krokiem do wykonania przez administratora systemu jest wprowadzenie ograniczeń. Dokonuje się tego dla każdego użytkownika osobno, najprościej przy pomocy polecenia *edquota*, które uruchamia edytor z czytelną tabelką, przypominającą raport polecenia *quota*. Jako argument polecenia podajemy nazwę użytkownika. Poniżej przedstawiono zaproponowane ograniczenia:

1	Disk quotas for user antek (uid 501):						
2	Filesystem	blocks	soft	hard	inodes	soft	hard
3	/dev/sda5	6756	10000	15000	48	0	0
4	/dev/sda3	472	5000	8000	23	1000	1500

Wartości zależą od dostępnych zasobów z uwzględnieniem zmiany liczby użytkowników systemu oraz dostępnych w danym systemie plików zasobów. Przykładowy system plików */home* nieco ponad 2 mln. bloków danych oraz 130 tys. i-węzłów. Przyjmując docelową liczbę użytkowników na 300 ustalamy ograniczenie miękkie dla bloków na 10 tys., co jest nieco powyżej wartości otrzymanej z podzielenia liczby dostępnych bloków przez liczbę użytkowników. Wynika to z faktu, że nie wszyscy użytkownicy będą w jednakowym stopniu eksploatowali zasoby, a jeśli ulegną one wyczerpaniu, to w systemie jest jeszcze wolne miejsce na innych dyskach. Wówczas system plików */home* można będzie zorganizować na partycji logicznej LVM je wykorzystującej. Ograniczenie twarde ustala się na 20–50% większe od ograniczenia miękkiego. Liczby i-węzłów nie limitujemy. W naszym systemie będzie przypadało średnio ok. 400 i-węzłów na użytkownika. Inaczej postępujemy w systemie plików */tmp*, gdzie istnieje możliwość tworzenia przez różne oprogramowanie dużej liczby małych plików.

Pozostała jeszcze kwestia ustalenia i zapisania limitu czasowego. Najczęściej spotykane wartości to 5 lub 7 dni. Wartości te ustalane są dla wszystkich użytkowników i grup w obrębie danego systemu plików. Służy do tego komenda *edquota* uruchomiona z opcją *-t*. Również w tym przypadku uruchamiany jest edytor, wraz z czytelną tabelką zawierającą informację o możliwych do użycia jednostkach oraz ustawione wartości domyślne:

1	Grace period before enforcing soft limits for users:		
2	Time units may be: days, hours, minutes, or seconds		
3	Filesystem	Block grace period	Inode grace period
4	/dev/sda5	7days	7days
5	/dev/sda3	7days	7days

Komenda *edquota* faktycznie uruchamia edytor, a po zakończeniu jego pracy prosty program sprawdzający poprawność zapisywanej informacji. Jeśli przykładowo pojawi się napis, którego postać nie jest liczbą w miejscu, w którym ma być wpisana liczba, lub użyto jednostek, które nie są akceptowalne, to komenda wypisze informacje o błędzie, a zmiany nie zostaną wprowadzone.

Mamy już ustalone wartości ograniczeń i limitów czasowych dla jednego użytkownika. Co z pozostałymi? Najczęściej wykorzystywanym sposobem jest kopiowanie ustawień jednego użytkownika (przyjętych za wzorcowe) na ustawienia następnego, a następnie ewentualna ich modyfikacja, jeśli zachodzi taka konieczność. W tym celu wykorzystujemy również komendę *edquota* z opcją **-p** podając jako pierwszy argument nazwę użytkownika którego ustawienia mają zostać skopiowane, a jako drugi nazwę użytkownika, któremu ustawienia te kopiujemy. W naszym przypadku ustawienia użytkownika *antek* skopiujemy użytkownikowi *jan*:

```
[root@messy ~]# edquota -p antek jan
```

Raport o wykorzystaniu zasobów systemów plików, których zasoby podlegają limitowaniu, w systemach linuxowych otrzymujemy dzięki poleceniu *repquota* uruchomionemu z opcją **-a**. W przypadku naszego systemu wygląda on następująco:

```
[root@messy ~]# repquota -a
*** Report for user quotas on device /dev/sda5
Block grace time: 7days; Inode grace time: 7days
      Block limits
User      used  soft  hard  grace
-----
root      --   35888      0      0          4      0      0
bin       --      4      0      0          1      0      0
jan       --    540  10000  15000       27      0      0
antek     --   6756  10000  15000       48      0      0

*** Report for user quotas on device /dev/sda3
Block grace time: 7days; Inode grace time: 7days
      Block limits
User      used  soft  hard  grace
-----
root      --   35912      0      0          6      0      0
jan       --    116   5000   8000       12  1000  1500
antek     --    472   5000   8000       23  1000  1500
```

Po stronie użytkownika *antek*, po przekroczeniu ograniczenia miękkiego, przykładowy raport polecenia *quota* będzie następujący:

```
[antek@messy tmp]$ cp /lib/libe* .
sda3: warning, user block quota exceeded.
[antek@messy tmp]$ quota -v
Disk quotas for user antek (uid 501):
  Filesystem  blocks  quota  limit  grace  files  quota  limit  grace
  /dev/sda5   6760   10000  15000      49      0      0
  /dev/sda3   5312*   5000   8000   7days    56   1000  1500
```

W systemie plików `/dev/sda3` użytkownik ma jeszcze dostępnych ok 2700 bloków danych. Musi jednak pamiętać, aby w ciągu 7-miu dni zwolnić część bloków tak, aby liczba wykorzystanych nie przekraczała ograniczenia miękkiego. W przeciwnym wypadku, pomimo nie osiągnięcia ograniczenia twardego nie będzie mógł wykorzystać potencjalnie dostępnych bloków. Ostrzeżenie (linia 2) pojawiło się, gdyż użytkownik był do systemu podłączony. Co jeśli przekroczenie limitu nastąpi wówczas, gdy użytkownik do systemu podłączony nie będzie? Administrator ma do dyspozycji program narzędziowy o nazwie *warnquota*. Jego działanie polega na przeglądaniu objętych reglamentacją systemów plików i wysyłanie wiadomości (maila) do użytkowników, którzy przekroczyli ograniczenie miękkie. Uruchomienie programu *warnquota* wykonuje się najczęściej z użyciem demona zegarowego *cron*. Zostanie ono omówione dokładnie w rozdziale poświęconym procesom.

Reglamentowanie zasobów systemów plików dla grup użytkowników w systemach linuksowych odbywa się w identyczny sposób, jak dla użytkowników indywidualnych. Przytoczone komendy wymagają użycia opcji **-g**, poprzedzającej nazwę grupy, gdyż domyślnie przyjmują jako argument nazwę użytkownika. Ograniczenia dla grupy obejmują wszystkich użytkowników, dla których grupa ta jest grupą podstawową. Jeśli grupą podstawową dla użytkowników *jan* oraz *antek* jest grupa *studenci*, to ograniczenia zasobów dla tej grupy definiujemy poleceniem:

```
1 [root@messy ~]# edquota -g studenci
```

Użytkownik może sprawdzić ograniczenia nałożone na grupy, do których należy poleceniem *quota* z opcjami **-vg**. Najistotniejsze są ustawienia dla grupy podstawowej, gdyż ona w znakomitej większości staje się właścicielem nowotworzonego pliku lub katalogu.

```
1 [antek@messy ~]$ quota -vg
2 Disk quotas for group antek (gid 502):
3   Filesystem  blocks    quota   limit   grace   files   quota   limit   grace
4   /dev/sda5      0         0       0         49        0         0
5   /dev/sda3      0         0       0        56        0         0
6 Disk quotas for group studenci (gid 503):
7   Filesystem  blocks    quota   limit   grace   files   quota   limit   grace
8   /dev/sda5   6760  2000000 2200000         0  20000  25000
9   /dev/sda3   5312  1500000 1600000         0  15000  18000
```

**Składnia i podstawowe opcje poleceń systemu** W przedstawionych powyżej przykładach wykorzystano kilka poleceń arbitralnie używając ich opcji. Dla uzupełnienia wiadomości, w dalszej części omówiono ich najczęściej wykorzystywane opcje. Polecenie *quotacheck*, które służy przeglądaniu wskazanych systemów plików w celu wyznaczenia ilości wykorzystywanych zasobów, tworzeniu, sprawdzaniu oraz naprawie plików bazy danych o wykorzystywanych zasobach. Jako argumentu wymaga podania punktu montowania lub nazwy urządzenia reprezentującego system plików. Pominięcie argumentu wymaga użycia opcji **-a**, co skutkuje przeglądnieniem wszystkich systemów plików zdefiniowanych w pliku `/etc/mtab`, zamontowanych z odpowiednimi opcjami. Z kilku powodów nie zaleca się uruchamiania polecenia *quotacheck* dla systemu plików, dla którego system kontyngentów jest włączony. Najczęściej wykorzystywanymi opcjami są:

- **-b, -backup** – opcja umożliwia stworzenie kopii zapasowej plików bazy danych wykorzystywanych zasobów, zanim nowa informacja o wykorzystanych zasobach zostanie zapisana.

- **-v, -verbose** – opcja wykorzystywana w dużych systemach plików, których zasoby są wykorzystywane przez dużą liczbę użytkowników. Opcja powoduje pojawianie się informacji o wykonywanych czynnościach. Spowalnia to jednak proces przeglądania systemu plików.
- **-d, -debug** – opcja włącza szczegółowy tryb raportowania.
- **-u, -user** – użycie opcji spowoduje weryfikację ograniczeń nałożonych na użytkowników indywidualnych w systemie plików, który pojawił się jako argument uruchomienia polecenia lub we wszystkich systemach plików zdefiniowanych w pliku */etc/mtab*.
- **-g, -group** – działanie identyczne jak opcji **-u** ale w odniesieniu do ograniczeń nałożonych na grupy użytkowników.
- **-c, -create-files** – opcja powoduje, że istniejące pliki bazy danych nie są czytane. Wyznaczane jest zużycie zasobów w systemach plików i dane te są zapisywane. Opcje wykorzystuje się, jeśli pliki nie istnieją, np. podczas inicjalizacji systemu.
- **-f, -force** – umożliwia wyznaczenie wykorzystania zasobów i zapis informacji do plików bazy danych przy włączonym systemie kontyngentów. Stosowanie opcji nie jest zalecane ze względu na możliwość powstania rozbieżności spowodowanych operacją synchronizacji systemu plików.
- **-M, -try-remount** – opcja pozwala na wyznaczanie ilości wykorzystywanych zasobów w systemie plików dostępnym do zapisu i odczytu, jeśli nie udało się go przemontować w tryb tylko do odczytu. Taki sposób działania jest dopuszczalny jedynie wówczas, gdy jesteśmy pewni, że żaden proces nie będzie zapisywał informacji podczas działania polecenia *quotacheck*.
- **-m, -no-remount** – użycie opcji będzie skutkowało tym, że na czas pracy polecenia system plików nie zostanie przemontowany w tryb tylko do odczytu. Opcję można stosować, jeśli jesteśmy pewni, że żaden proces w trakcie działania polecenia nie wykona operacji zapisu informacji do analizowanego systemu plików.
- **-i, -interactive** – domyślnie polecenie *quotacheck* kończy swoje działanie po wystąpieniu pierwszego błędu. W tym trybie użytkownik jest pytany o kontynuację.
- **-n, -use-first-dquot** – istnieje możliwość, że pliki bazy danych zostaną uszkodzone. Najczęściej spotykane uszkodzenie polega na tym, że istnieje w nich więcej niż jeden rekord opisujący wykorzystanie zasobów przez danego użytkownika lub grupę. Domyślnie w takiej sytuacji wykonanie polecenia *quotacheck* kończy się błędem. Użycie opcji powoduje, że wykorzystywana jest informacja z pierwszego rekordu bez zakończenia działania polecenia.
- **-F, -format=nazwa\_formatu** – opcja pozwala na sprawdzenie i ustalenie formatu plików bazy danych systemu kontyngentów. Polecenie *quotacheck* nie dokonuje wówczas autodekacji. Możliwe nazwy formatów to: *vfsold* – wersja 1, *vfsv0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-a, -all** – opcja wymusza sprawdzenie każdego, zdefiniowanego w pliku */etc/mtab* systemu plików, zamontowanego ze stosownymi opcjami. Sprawdzeniu nie podlegają sieciowe systemy plików.
- **-R, -exclude-root** – użyta z opcją **-a** powoduje, że sprawdzone zostaną wszystkie systemy plików zdefiniowane w pliku */etc/mtab* za wyjątkiem głównego (/) systemu plików.

Polecenie *quotaon* służy do zdefiniowania w systemie jednego lub więcej systemu plików, w którym system kontyngentów jest aktywny. Dla poprawnej inicjalizacji systemu kontyngentów wymagane jest istnienie w katalogu głównym systemu plików, plików bazy danych, których nazwy zależne są od wersji. Dla wersji 1 są to pliki *quota.user* dla przechowywania informacji o zasobach wykorzystywanych przez użytkowników oraz *quota.group* dla grup użytkowników. W wersji 2 pliki te nazywają się odpowiednio *aquota.user* oraz *aquota.group*. Konwencja ta dotyczy wszystkich rodzajów systemów plików za wyjątkiem systemu XFS, który informacje systemu kontyngentów przechowuje w swoich metadanych. Polecenie *quotaon* wymaga podania nazwy systemu plików, dla którego ma zostać uaktywniony system kontyngentów. Argument można pominąć stosując opcję **-a**, co spowoduje uaktywnienie systemu dla wszystkich systemów plików wymienionych w pliku */etc/mtab*, zamontowanych ze stosownymi opcjami, za wyjątkiem sieciowych systemów plików. Polecenie *quotaon* nie działa dla systemu plików XFS, dla którego system kontyngentów uruchamia się w momencie montowania, używając odpowiednich opcji (patrz podpunkt 3.4.5).

Spśród najczęściej wykorzystywanych opcji polecenia *quotaon* należy wyróżnić:

- **-F, -format=nazwa\_formatu** – opcja powoduje, że informacje systemu kontyngentów zapisywane są w wyspecyfikowanym formacie. Możliwe nazwy formatów to: *vsold* – wersja 1, *vs0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-a, -all** – opcja uruchamia system kontyngentów dla wszystkich zdefiniowanych w pliku */etc/mtab* systemów plików, które zostały zamontowane automatycznie (opcja *noauto*) z opcjami systemu kontyngentów (*usrquota, grpquota*).
- **-v, -verbose** – polecenie raportuje każdy system plików, dla którego system kontyngentów został włączony.
- **-u, -user** – jest to opcja domyślna, włączająca zarządzanie ograniczeniami dla użytkowników.
- **-g, -group** – opcja pozwala włączyć zarządzanie ograniczeniami na zasoby wykorzystywane przez grupy użytkowników.
- **-p, -print-state** – użycie opcji spowoduje wypisanie informacji o stanie systemu kontyngentów.
- **-f, -off** – opcja powoduje, że polecenie *quotaon* działa jak *quotaoff*.

Polecenie *quotaoff* służy do wyłączania systemu kontyngentów dla systemu plików, którego punkt montowania lub nazwa reprezentującego urządzenia pojawiła się jako argument wywołania. Pominięcie argumentu wymusza użycie opcji **-a**, co skutkuje wyłączeniem systemu kontyngentów dla systemów plików zdefiniowanych w pliku */etc/mtab* z odpowiednimi opcjami montowania, za wyjątkiem sieciowych systemów plików. Wyłączanie systemu kontyngentów stosuje się z kilku powodów. Jednym z nich może być konieczność uruchomienia polecenia *quotacheck* dla odtworzenia uszkodzonych plików bazy danych systemu kontyngentów. Polecenie *quotaoff* posiada kilka opcji, do których należą:

- **-F, -format=nazwa\_formatu** – opcja informuje o formacie zapisu danych systemu kontyngentów. Możliwe nazwy formatów to: *vsold* – wersja 1, *vs0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-a, -all** – opcja wyłącza system kontyngentów dla wszystkich zdefiniowanych w pliku */etc/mtab* systemów plików, które zostały zamontowane automatycznie (opcja *noauto*) z opcjami systemu kontyngentów (*usrquota, grpquota*).



- **-v, -verbose** – polecenie raportuje każdy system plików, dla którego system kontyngentów został wyłączony.
- **-u, -user** – opcja pozwala wyłączyć zarządzanie ograniczeniami dla użytkowników.
- **-g, -group** – opcja wyłącza zarządzanie ograniczeniami na zasoby wykorzystywane przez grupy użytkowników.
- **-p, -print-state** – użycie opcji spowoduje, że zamiast uczynić system kontyngentów nieaktywnym, zostanie wypisana informacja o jego stanie.
- **-x, -xfs-command delete** – opcja działa efektywnie jedynie w systemie plików XFS. Pozwala zwolnić miejsce zajmowane w strukturach metadanych na przechowywanie danych systemu kontyngentów. Użycie opcji wymaga uprzedniego wyłączenia systemu kontyngentów.
- **-x, -xfs-command enforce** – podobie jak poprzednia, opcja ta działa efektywnie w systemach plików XFS. Efektem jej działania jest uruchomienie systemu kontyngentów polegające na wyznaczaniu ilości wykorzystanych zasobów.

Polecenie *edquota* to edytor ograniczeń wykorzystania zasobów systemu plików nakładanych na użytkownika indywidualnego lub grupę użytkowników. Argumentem wywołania polecenia może być nazwa co najmniej jednego użytkownika lub grupy użytkowników. Jeśli zamiast nazwy pojawi się liczba, to jest ona traktowana odpowiednio jako numer identyfikacyjny użytkownika lub grupy. Po uruchomieniu polecenia, dla każdego użytkownika lub grupy tworzony jest w katalogu */tmp* tekstowy plik tymczasowy, zawierający definicję nałożonych ograniczeń. Plik jest edytowany przez edytor zdefiniowany zmienną środowiskową *EDITOR*. W domyślnej konfiguracji jest to edytor *vi*. Stąd podczas edycji obowiązują zasady pracy danego edytora. Zakończenie pracy bez zapisu edytowanego pliku spowoduje, że nie zostaną podjęte żadne, dalsze czynności. Zapisanie wprowadzonych zmian skutkuje podjęciem próby zapisania ich w plikach bazy danych systemu kontyngentów. Uruchamiany jest proces parsujący zawartość pliku i jeśli znaleziony zostanie błąd (np. napis zamiast liczby), to pojawia się stosowny komunikat. W przeciwnym przypadku ustawienia zostają zapisane w plikach bazy danych.

Podstawowe opcje polecenia *edquota* to:

- **-r, -remote** – opcja pozwala poleceniu *edquota* na edytowanie ograniczeń systemu kontyngentów w systemie zdalnym. Wymagana jest kompilacja programu ze wsparciem dla mechanizmu zdalnego wywołania procedury (*Remote Procedure Call*). Funkcjonalnie opcja odpowiada opcji **-n**.
- **-u, -user** – opcja domyślana, umożliwiająca edycję ograniczeń nałożonych na użytkowników indywidualnych.
- **-g, -group** – polecenie uruchamia się z tą opcją w celu edycji ograniczeń nałożonych na grupy użytkowników.
- **-p, -prototype=protoname** – opcje stosuje się w celu skopiowania ograniczeń ustawionych dla użytkownika, którego nazwa lub numer identyfikacyjny pojawiła się jako wartość opcji użytkownikom, których nazwy lub numery identyfikacyjne stanowią argumenty uruchomienia polecenia. Ten sam mechanizm jest dostępny dla kopiowania ustawień narzuconych na grupy użytkowników.

- **-F, -format=nazwa\_formatu** – opcja umożliwia edycję ograniczeń zapisanych w różnych formatach. Możliwe nazwy formatów to: *ufsold* – wersja 1, *ufsv0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-f, -filesystem system\_plików** – wartością opcji jest punkt montowania lub nazwa urządzenia reprezentującego system plików, dla którego będą edytowane ograniczenia systemu kontyngentów.
- **-t, -edit-period** – opcja umożliwia edycję limitu czasu dla ograniczenia miękkiego.
- **-T, -edit-times** – opcja pozwala na modyfikację limitu czasu dla pojedynczego użytkownika lub grupy.

Odpowiednikiem polecenia *edquota*, pracującym jednak w oparciu o linię komend jest polecenie *setquota*. Umożliwia ono zarówno bezpośrednie ustawianie ograniczeń na wykorzystanie zasobów systemu i limitów czasowych, jak również ustawianie i kopiowanie wzorcowych limitów czasowych na ograniczenie miękkie. Posiada możliwość pracy w trybie wsadowym, czytając ustawienia ze standardowego wejścia. Bogata funkcjonalność prowadzi jednak do relatywnie dużej złożoności składniowej. W trybie interaktywnym wymagane jest podanie nazwy użytkownika lub grupy, wartości ograniczeń miękkiego i twardego nałożonego na bloki danych, wartości ograniczeń twardego i miękkiego nałożonego na i-węzły oraz punktu montowania lub nazwy pliku reprezentującego urządzenie, na którym znajduje się system plików. Jeśli w miejscu nazwy użytkownika lub grupy pojawi się liczba, jest ona traktowana jako numer identyfikacyjny. Użycie opcji **-a** powoduje, że ograniczenia zostaną wprowadzone we wszystkich systemach plików zdefiniowanych w pliku */etc/mtab*, zamontowanych z odpowiednimi opcjami z wyjątkiem sieciowych systemów plików. W takim przypadku ogólna składnia polecenia jest następująca:

```
setquota [opcja] nazwa_uzytkownika_lub_grupy
miękkie_ograniczenie_na_bloki twarde_ograniczenie_na_bloki
miękkie_ograniczenie_na_i_węzły twarde_ograniczenie_na_i_węzły
-a | system_plików
```

Polecenie *setquota* umożliwia również ustalanie ograniczeń przez kopiowanie ograniczeń nadanych dowolnemu użytkownikowi lub grupie, traktowanych jako wzorcowe. W tym trybie, po opcji **-p**, podawana jest nazwa użytkownika lub grupy, ustawienia których mają zostać skopiowane.

Istnieje również możliwość ustalania limitów czasowych dla ograniczeń miękkich, całościowo oraz dla wybranych użytkowników lub grup. Opcje pracy polecenia pokrywają się z omówionymi wcześniej opcjami polecenia *edquota*. Najczęściej wykorzystywanymi są:

- **-r, -remote** – opcja pozwala poleceniu *edquota* na edytowanie ograniczeń systemu kontyngentów w systemie zdalnym. Wymagana jest obsługa mechanizmu zdalnego wywołania procedury (*Remote Procedure Call*).
- **-F, -format=nazwa\_formatu** – opcja informuje o formacie zapisu danych systemu kontyngentów. Możliwe nazwy formatów to: *ufsold* – wersja 1, *ufsv0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-u, -user** – opcja domyślana, umożliwiająca edycję ograniczeń nałożonych na użytkowników indywidualnych.
- **-g, -group** – polecenie uruchamia się z tą opcją w celu edycji ograniczeń nałożonych na grupy użytkowników.

- **-p, -prototype** = *protoname* – opcje stosuje się w celu skopiowania ograniczeń ustawionych dla użytkownika, którego nazwa lub numer identyfikacyjny pojawiła się jako wartość opcji użytkownikowi, którego nazwa lub numer identyfikacyjny pojawił się jako argument uruchomienia polecenia. Ten sam mechanizm jest dostępny dla kopiowania ustawień narzuconych na grupy użytkowników.
- **-b, -batch** – opcja jest wykorzystywana do przełączania trybu pracy polecenia *setquota* na wsadowy. Wówczas, jako argument podawany jest punkt montowania lub nazwa pliku urządzenia na którym założono system plików. Argument można zastąpić opcją **-a**. Nazwy lub numery identyfikacyjne użytkowników lub grup oraz nałożone na nich ograniczenia czytane są w trybie wsadowym ze standardowego wejścia, w formacie: *nazwa\_użytkownika\_lub\_grupy miękke ograniczenie\_na\_bloki twarde ograniczenie\_na\_bloki miękke ograniczenie\_na\_i\_węzły twarde ograniczenie\_na\_i\_węzły*.
- **-t, -edit-period** – opcja umożliwia edycję limitu czasu dla ograniczenia miękkiego. Jednostką są sekundy.
- **-T, -edit-times** – opcja pozwala na modyfikację limitu czasu dla pojedynczego użytkownika lub grupy. Jednostką są sekundy. Może pojawić się napis *unset*, jeśli nie chcemy narzucać limitów.
- **-a, -all** – opcję wykorzystujemy wówczas, gdy ustawienia mają zostać wprowadzone we wszystkich systemach plików zdefiniowanych w pliku */etc/mtab*, montowanych z opcjami dla systemu kontyngentów (*usrquota, grpquota*) z wyjątkiem sieciowych systemów plików.

Polecenie *quota* umożliwia uzyskanie informacji o wykorzystywanych zasobach systemów plików, ustawionych dla nich ograniczeniach oraz limitach czasowych. Domyślnie wypisywane są dane dotyczące bieżącego użytkownika. Zwykły użytkownik nie może uzyskać informacji dotyczących innych użytkowników, jak również grup, do których nie należy. Do podstawowych opcji polecenia *quota* w tej dystrybucji należą:

- **-F, -format** = *nazwa\_formatu* – opcja umożliwia określenie formatu, w którym przechowywane są informacje systemu kontyngentów. Możliwe wartości opcji to: **vfsold** dla wersji 1, **vfsv0** dla wersji 2, **rpc** dla sieciowych systemów plików NFS, **xf**s jeśli system kontyngentów działa w systemie plików XFS.
- **-g, -group** – opcja pozwala na wypisanie informacji dotyczącej grup, do których należy użytkownik. Informacje dla wybranej grupy można uzyskać podając jej nazwę jako wartość opcji.
- **-u, -user** – opcja domyślna, powoduje wypisanie informacji o zasobach wykorzystywanych przez bieżącego użytkownika. Wartością opcji może być nazwa użytkownika.
- **-v, -verbose** – opcja powoduje wypisanie informacji o wykorzystanych zasobach w systemach plików, dla których nie zostały określone wartości limitów i ograniczeń czasowych, ale system kontyngentów jest aktywny.
- **-s, -human-readable** – opcja służy formatowaniu informacji o wykorzystanych blokach dyskowych z wykorzystaniem jednostek innych niż bloki dyskowe. Stąd zamiast długich ciągów cyfr oznaczających liczbę bloków dyskowych pojawiają się krótkie liczby z jednostkami MB, GB, TB. W innych komendach systemów tej dystrybucji takie formatowanie zapewnia opcja **-h**.

- **-p, -raw-grace** – powoduje, że informacja o ilości czasu, który pozostał po przekroczeniu ograniczenia miękkiego jest podawana w sekundach.
- **-i, -no-autofs** – opcja powoduje, że polecenie ignoruje punkty montowania automontera.
- **-l, -local-only** – opcja powoduje ograniczenie dostarczanej przez polecenie informacji jedynie do lokalnych systemów plików.
- **-A, -all-nfs** – opcja powoduje dostarczenie informacji o wszystkich sieciowych systemach plików.
- **-q, -quiet** – użycie opcji spowoduje ograniczenie dostarczanej informacji jedynie do tych systemów plików, w których zostało przekroczone którekolwiek z ograniczeń.
- **-Q, -quiet-refuse** – działanie opcji dotyczy sieciowych systemów plików i powoduje, że nie zostanie wypisany komunikat o błędzie jeśli połączenie z procesem systemu kontyngentów działającym na zdalnym systemie nie było możliwe.
- **-w, -no-wrap** – opcja powoduje, że linie wydruku nie zostaną podzielone, nawet w przypadku długiej nazwy urządzenia, na którym dany system plików się znajduje. Jest to istotne, jeśli dostarczone dane mają być następnie przetwarzane przez skrypt.

Informacji o wykorzystaniu zasobów w systemie plików oraz ustawionych ograniczeniach dostarcza polecenie *repquota*. Dostarcza ono informacje o limitach czasowych dla ograniczeń miękkich obowiązujące w danym systemie plików, a następnie w formie tabelki informacje o użytkownikach, których pliki wykorzystują zasoby danego systemu plików w postaci: symbolicznego oznaczenia przekroczenia miękkich ograniczeń (+ oznacza, że limit został przekroczony, – limit nie został przekroczony), blokach danych i i-węzłów z wyszczególnieniem wykorzystanych, ograniczenia miękkiego, ograniczenia twardego oraz wykorzystania limitu czasowego w przypadku przekroczenia limitu czasowego. W trybie gadatliwym pojawia się również statystyka. Obejmuje ona m.in. liczbę użytkowników, którzy wykorzystują zasoby monitorowanych systemów plików.

Pośród licznych opcji polecenia *repquota* najczęściej wykorzystuje się:

- **-a, -all** – opcja umożliwia generowanie raportu o wykorzystaniu zasobów i stanu systemu kontyngentów dla systemów plików zdefiniowanych w pliku */etc/mstab*, zamontowanych z odpowiednimi opcjami. Opcja zastępuje argument.
- **-v, -verbose** – przełącza sposób generowania raportu w tryb gadatliwy. W praktyce raport zostaje uzupełniony o proste statystyki.
- **-c, -batch-translations** – to opcja domyślna, która powoduje tłumaczenie numerów identyfikacyjnych użytkowników i grup na ich nazwy. Opcja pozwala na szybkie działanie polecenia, jeśli plikami zawierającymi definicję użytkowników i grup są lokalne pliki */etc/passwd* oraz */etc/group*. Konieczność dostępu do baz danych użytkowników, w szczególności zdalnych, znacznie wydłuża czas pracy.
- **-C, -no-batch-translations** – opcja stosowana przy tłumaczeniu numerów identyfikacyjnych użytkowników i grup na nazwy jeśli ich definicje przechowywane są poza plikami */etc/passwd* oraz */etc/group*.
- **-t, -truncate-names** – opcja służy formatowaniu nazw użytkowników i grup poprzez skrócenie ich do pierwszych 9-ciu znaków.

- **-n, -no-names** – użycie opcji znacznie przyspiesza działanie polecenia *repquota*, gdyż powoduje, że numery identyfikacyjne użytkowników i grup użytkowników nie będą konwertowane na odpowiadające im nazwy.
- **-s, -human-readable** – opcja powoduje wypisanie informacji o wykorzystanych zasobach w formacie „czytelny dla człowieka”. Stąd, zamiast długich ciągów cyfr oznaczających liczbę zajętych bloków danych pojawiają się krótkie liczby z jednostkami MB, GB, TB. Zwykle w systemach uniksowych taki sposób formatowania zapewnia opcja **-h**.
- **-p, -raw-grace** – opcja umożliwia raportowanie momentu przekroczenia limitu na ograniczenie miękkie w postaci liczby sekund, które upłyną od godziny 0, 1 stycznia 1970r. Sposób ten jest wykorzystywany, jeśli dane mają być następnie procesowane.
- **-i, -no-autofs** – opcja powoduje, że raport nie jest generowany dla systemów plików montowanych programem automontera. Więcej informacji na ten temat znaleźć można w tomie poświęconym konfiguracjom sieciowym.
- **-F, -format=nazwa\_formatu** – opcja informuje o formacie zapisu danych systemu kontyngentów. Możliwe nazwy formatów to: *ufsold* – wersja 1, *ufsv0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-u, -user** – opcja ogranicza raport jedynie do użytkowników indywidualnych.
- **-g, -group** – opcja umożliwia uzyskanie raportu jedynie dla grup użytkowników.

Jeśli miękkie ograniczenie zostanie przekroczone w czasie, w którym użytkownik nie będzie podłączony do systemu, to bez użycia polecenie *quota* nie dowie się o tym fakcie. W systemach linuksowych możliwe jest cykliczne uruchamianie procesu, który przegłądnie lokalne systemy plików, w których działają systemy kontyngentów i wyśle pocztą elektroniczną stosowną informację użytkownikom, którzy przekroczyli ograniczenie miękkie. Możliwości takie udostępnia polecenie *warnquota*, którego cykliczne uruchamianie, jak wspomniano, realizuje się przez proces demona zegarowego *cron*.

Polecenie posiada plik konfiguracyjny o nazwie *warnquota.conf*, znajdujący się w katalogu */etc*. Jest to plik tekstowy. Znaki *#* lub *;* są symbolami komentarza, który zaczyna się tym znakiem, a kończy znakiem przejścia do nowej linii. Pozostałe linie mają ogólną postać: *NAZWA\_ZMIENNEJ=WARTOŚĆ*. Fragment pliku przedstawiono poniżej:

```

1 .....
2 # values can be quoted:
3 MAIL_CMD      = "/usr/sbin/sendmail -t"
4 FROM          =
5 # but they don't have to be:
6 SUBJECT       = NOTE: You are exceeding your allocated disk space limits
7 CC_TO        = "root@localhost"
8 .....
9 # Text in the beginning of the mail (if not specified, default text is used)
10 # This way text can be split to more lines
11 # Line breaks are done by '|' character
12 # The expressions %i, %h, %d, and %% are substituted for user/group name,
13 # host name, domain name, and '%' respectively. For backward compatibility
14 # %s behaves as %i but is deprecated.
15 MESSAGE      = Your disk usage has exceeded the agreed limits\

```

```

16 on this server|Please delete any unnecessary files on following filesystems:|
17 .....

```

Jak widać, zdefiniować można m.in. program, którym list elektroniczny zostanie wysłany (*MAIL\_CMD*), pochodzenie (*FROM*), nagłówek informacji (*SUBJECT*), oraz do kogo zostanie przesłana kopia (*CC\_TO*). Treść listu (*MESSAGE*) może zostać zapisana w kilku liniach. Znakiem kontynuacji jest \. Można wykorzystywać symbole formatowania. Przykładowo %i jest zastępowany nazwą użytkownika lub grupy, %h nazwą hosta, %d nazwą domeny.

W przypadku przekroczenia ograniczenia miękkiego przez użytkownika indywidualnego, adresat informacji wysyłanej przez polecenie *warnquota* jest jednoznacznie określony. Jednak w przypadku ograniczeń nałożonych na grupę użytkowników, przekroczenie ograniczenia miękkiego nie spowoduje wysłania informacji do wszystkich użytkowników, którzy należą do tej grupy. Informacja zostanie wysłana do użytkownika, który został określony jako zarządca systemu kontyngentów dla danej grupy użytkowników. Plikiem konfiguracyjnym jest w tym przypadku plik *quotagrpadmins* znajdujący się w katalogu */etc*. Jest to plik tekstowy, w którym w jednej linii znajduje się definicja administratora dla jednej grupy. Podobnie jak w innych plikach konfiguracyjnych, dopuszcza się komentarze linijkowe. Komentarz zaczyna się znakiem #, a kończy znakiem przejścia do nowej linii. Linie definicyjne zawierają nazwę grupy oddzieloną znakiem : od nazwy użytkownika, który jest jej administratorem. Administrator grupy dla systemu kontyngentów nie musi należeć do systemowej grupy użytkowników. Przykładowy plik zamieszczono poniżej:

```

1 #
2 # This is a sample groupadmins file (/etc/quotagrpadmins)
3 #
4 # Comments begin with hash in the beginning of the line
5
6 # In this file you specify users responsible for space used by the group
7 studenci:jan

```

Polecenie *warnquota* wykorzystuje jeszcze jeden plik konfiguracyjny. To plik *quotatab* z katalogu */etc*. Jest to plik tekstowy zawierający opis urządzeń na których znajduje się dany system plików. Chodzi o to, że zwykły użytkownik, nie znając polecenia *df*, może nie wiedzieć o który system plików chodzi. Tu istnieje możliwość jego opisu. Plik dopuszcza komentarz na zasadach takich, jak dla pozostałych plików konfiguracyjnych. Linie aktywne zawierają opis systemu plików oraz nazwę urządzenia, na którym system plików został założony oddzielone znakiem :. Przykład pliku zamieszczono poniżej:

```

1 #
2 # This is sample quotatab (/etc/quotatab)
3 # Here you can specify description of each device for user
4 #
5 # Comments begin with hash in the beginning of the line
6
7 # Description
8 /dev/sda5 : Home directory
9 /dev/sda3 : /tmp

```

Do najczęściej wykorzystywanych opcji polecenia *warnquota* należą:

- **-F, -format**=*nazwa\_formatu* – opcja informuje o formacie zapisu danych systemu kontyngentów. Możliwe nazwy formatów to: *ufsold* – wersja 1, *ufsv0* – wersja 2, *rpc* – dla sieciowych systemów plików, *xfs* – dla systemu plików XFS.
- **-q, -quota-tab**=*plik\_quotatab* – wartością opcji jest ścieżka dostępu do pliku zawierającego opis urządzeń, alternatywnego do pliku */etc/quotatab*.
- **-c, -config**=*configfile* – opcja umożliwia podanie nazwy pliku zawierającego konfigurację polecenia *warnquota*, różnego od pliku */etc/warnquota.conf*.
- **-a, -adminsfile**=*adminsfile* – opcja umożliwia podanie nazwy pliku zawierającego nazwy użytkowników będących administratorami grup dla systemu kontyngentów, różnego od pliku */etc/warnquota.conf*.
- **-u, -user** – to opcja domyślna, w wyniku działania której polecenie sprawdza stan systemu kontyngentów dla użytkowników indywidualnych.
- **-g, -group** – polecenie *warnquota* będzie działało w odniesieniu do ograniczeń nałożonych na grupy użytkowników.
- **-s, -human-readable** – opcja pozwala formatować informacje o wykorzystanych zasobach w sposób czytelny dla człowieka (jednostki zamiast liczb zużytych bloków dyskowych).
- **-i, -no-autofs** – opcja wyłącza sprawdzanie plików montowanych programem automontera.
- **-d, -no-details** – opcja skraca informację wysyłaną do użytkowników do ostrzeżenia o przekroczeniu limitu. Informacja o wykorzystaniu zasobów jest pomijana.

### Quota w systemie FreeBSD

W systemach z rodziny BSD administrowanie systemem kontyngentów jest nieco bardziej skomplikowane. Domyślna konfiguracja jądra systemu nie zakłada bowiem wykorzystywania systemu kontyngentów. Stąd konieczne staje się skompilowanie nowego jądra. Choć brzmi to groźnie, w rzeczywistości nie ma się czego obawiać. W tym rozdziale pokażemy, jak utworzyć nowe jądro zawierające wymagane funkcjonalności, zaś w tomie trzecim pojawi się stosowna teoria.

Plikiem konfiguracyjnym podstawowej wersji jądra jest plik o nazwie *GENERIC* znajdujący się w tym systemie w katalogu */usr/src/sys/i386/conf* (różnice nazwy wynikają z architektury procesora, gdyż od niej zależy nazwa przedostatniego katalogu). Jest to plik tekstowy. Dla zachowania starej konfiguracji jądra skopiujemy go na inny, przykładowo o nazwie *MYKERNEL*. Dodanie do jądra systemu obsługi kontyngentów dyskowych, wymaga umieszczenia w pliku konfiguracyjnym następującej linii:

```
1 options          QUOTA
```

Następnie przechodzimy do katalogu */usr/src* i kompilujemy poprawione jądro systemu operacyjnego:

```
1 -su-4.0# cd /usr/src
2 -su-4.0# make buildkernel KERNCONF=MYKERNEL
```

Dobłą praktyką jest zachowanie starego, działającego jądra systemu. Zazwyczaj nowe jądro powstaje w wyniku kilku iteracji, a często zdarza się, że pierwsze utworzone nie uruchomi się. Dlatego też, przed instalacją nowego jądra kopiujemy składniki starego. Aktualne jądro systemu znajduje się w katalogu */boot/kernel*. Mamy zatem:

```
1 -su-4.0# cp -R /boot/kernel /boot/kernel.good
2 -su-4.0# make installkernel KERNCONF=MYKERNEL
```

Nowe jądro systemu zostanie zainstalowane w katalogu */boot/kernel*. Wykorzystywane dotychczas zostanie przeniesione do katalogu */boot/kernel.old*. Dodatkowa kopia znajduje się w katalogu */boot/kernel.good*. Teraz należy ponownie uruchomić system. Jeśli pojawią się problemy z jego inicjalizacją, to należy zainicjalizować system w oparciu o poprzednie jądro, sprawdzić w logach systemu komunikaty o błędach, na ich podstawie poprawić konfigurację jądra i dokonać jego ponownej kompilacji i instalacji. Jak wspomniano, procedury te zostaną dokładnie omówione w tomie drugim.

Kolejnym krokiem jest zapewnienie zamontowania właściwych systemów plików z opcjami umożliwiającymi działanie systemu kontyngentów dla użytkowników indywidualnych i/lub grup użytkowników. Aby montowanie miało charakter trwały, odpowiednia informacja musi zostać zapisana w pliku */etc/fstab*. Następnie, w katalogach głównych systemów plików dla których ma funkcjonować system kontyngentów należy utworzyć pliki bazy danych przechowujące informacje o wykorzystaniu zasobów. Od tej pory rozpoczyna się zarządzanie utworzoną strukturą, co polega na nakładaniu ograniczeń na wykorzystanie zasobów systemu plików na użytkowników indywidualnych i grupy użytkowników, zmianę wartości ograniczeń i limitów czasowych oraz, jeśli zachodzi taka konieczność, wyłączanie, włączanie i aktualizacja zawartości plików bazy danych systemu kontyngentów.

**Przykład działania systemu** Zasady administrowania systemem kontyngentów w systemie FreeBSD poznamy również na przykładzie. W systemie mamy dwa systemy plików, zasoby których mogą być wykorzystywane przez każdego użytkownika systemu. Są one montowane w katalogach */home* oraz */tmp*. Pierwszym krokiem jest zamontowanie tych systemów plików z opcjami, które wymuszają obsługę systemu kontyngentów. Aby zmiany te miały charakter trwały, tzn. system kontyngentów działał po przeładowaniu systemu operacyjnego, opcje te dopisujemy w pliku definiującym systemy plików danego systemu operacyjnego, a więc */etc/fstab*. Istnieje możliwość kontrolowania zużycia zasobów przez użytkownika indywidualnego, co uzyskujemy montując system z opcją *userquota*, oraz zasobów zużytych przez grupę użytkowników, co załatwia opcja *groupquota*. Zawartość przykładowego pliku jest teraz następująca:

#	Device	Mountpoint	FStype	Options	Dump	Pass#
2	/dev/ad0s1b	none	swap	sw	0	0
3	/dev/ad0s1a	/	ufs	rw	1	1
4	/dev/ad0s1e	/home	ufs	rw,acls,userquota,groupquota	2	2
5	/dev/ad0s1f	/tmp	ufs	rw,userquota,groupquota	2	2
6	/dev/ad0s1d	/usr	ufs	rw	2	2
7	/dev/ad0s1g	/var	ufs	rw	2	2
8	/dev/acd0	/cdrom	cd9660	ro,noauto	0	0

Opcje montowania dla przedmiotowych systemów plików pojawiły się w liniach 4 oraz 5. Ponieważ nasz system operacyjny pracuje, dokonamy przemontowania jedynie tych systemów plików, których opcje uległy zmianie. Mamy zatem:



```
1 -su-4.0# mount -a
```

Następnie w katalogach głównych systemów plików należy utworzyć pliki bazy danych przechowujące informacje o wykorzystaniu zasobów systemów plików przez użytkowników i grupy użytkowników. Służy do tego komenda *quotacheck*:

```
1 -su-3.2# quotacheck -ugv /home
2 *** Checking user and group quotas for /dev/ad0s1e (/home)
3 -su-3.2# ls -l /home/qu*
4 -rw-r----- 1 root operator 32096 Sep 17 11:27 /home/quota.group
5 -rw-r----- 1 root operator 32096 Sep 17 11:27 /home/quota.user
6 -su-3.2# quotacheck -ugv /tmp
7 *** Checking user and group quotas for /dev/ad0s1f (/tmp)
8 -su-3.2# ls -l /tmp/qu*
9 -rw-r----- 1 root operator 192 Sep 17 11:29 /tmp/quota.group
10 -rw-r----- 1 root operator 32 Sep 17 11:29 /tmp/quota.use
```

Teraz, poleceniem *quotaon* uruchamiamy systemy kontyngentów dla odpowiednich systemów plików. Jeśli w zamyśle mają to być wszystkie systemy plików, dla których w pliku */etc/fstab* zdefiniowano odpowiednie opcje montowania, wystarczy użyć opcji *-a*. Jeśli system kontyngentów ma zostać uruchomiony dla wybranych systemów plików, to punkty ich montowania pojawiają się jako argumenty komendy. W naszym przypadku zachodzi przypadek pierwszy:

```
1 -su-3.2# quotaon -a
```

System kontyngentów działa. Można zatrzymać jego działanie poleceniem *quotaoff*. Istotne jest także to, że w obecnej konfiguracji nie wznowi on działania po ponownym uruchomieniu systemu operacyjnego. W żadnym pliku konfiguracyjnym nie została bowiem zapisana informacja o jego uruchamianiu podczas inicjalizacji systemu operacyjnego. W celu rozwiązania tego problemu, w pliku */etc/rc.conf* umieszczamy następującą linię:

```
1 enable_quotas="YES"
```

Wpis ten spowoduje inicjalizację systemu kontyngentów dla odpowiednich systemów plików ale z jednoczesnym uaktualnieniem zawartości baz danych wykorzystanych zasobów. Spowoduje to wydłużenie czasu inicjalizacji systemu. Wykonywanie tej czynności, odpowiadającej wykonaniu z linii komend polecenia *quotacheck* wyłącza wpis, umieszczony oczywiście w pliku */etc/rc.conf*.

```
1 check_quotas="NO"
```

Dotychczasowa konfiguracja systemu kontyngentów ogranicza jego działanie jedynie do zliczania zasobów wykorzystywanych przez użytkowników i grupy w odpowiednich systemach plików. Każdy użytkownik może sprawdzić ile bloków danych oraz i-węzłów wykorzystują pliki, których jest właścicielem indywidualnym. Podobnie jak w innych systemach operacyjnych służy do tego polecenie *quota* uruchomione z opcją *-v*:

1	Disk quotas for user jan (uid 1002):								
2	Filesystem	usage	quota	limit	grace	files	quota	limit	grace
3	/home	18064	0	0		69	0	0	
4	/tmp	14448	0	0		21	0	0	

Format uzyskanej informacji jest identyczny, jak dla komendy *quota* z systemów rodziny RedHat. Jedyną różnicą, to opis systemu plików, którym w tym przypadku jest punkt montowania systemu plików, a nie nazwa urządzenia, na którym on się znajduje. Jak widać użytkownik nie ma narzuconych żadnych ograniczeń. Jak chodzi o limity czasowe, to przy pomocy komendy *quota* nie możemy tego określić, jeśli nie zostało przekroczone ograniczenie miękkie. Kolejnym krokiem do wykonania przez administratora systemu jest określenie wartości ograniczeń oraz limitów czasowych indywidualnie dla każdego użytkownika. Do dyspozycji mamy polecenie *edquota*, które w najprostszym przypadku uruchamiamy z argumentem będącym nazwą lub numerem identyfikacyjnym użytkownika. Polecenie uruchamia edytor z plikiem tymczasowym zawierającym wartości ograniczeń. W konfiguracji domyślnej jest to edytor *vi*. Decyduje o tym wartość zmiennej środowiskowej *EDITOR*. W przypadku systemu FreeBSD jego format jest całkowicie inny niż format wypisywania informacji przez polecenie *quota*. Poniżej przedstawiono proponowane limity:

```

1 Quotas for user jan:
2 /home: kbytes in use: 18064, limits (soft = 25000, hard = 30000)
3     inodes in use: 69, limits (soft = 5000, hard = 8000)
4 /tmp: kbytes in use: 14448, limits (soft = 8000, hard = 10000)
5     inodes in use: 21, limits (soft = 3000, hard = 6000)

```

Wartości te ustalono biorąc pod uwagę dostępne zasoby, możliwość ich rozbudowy oraz aktualną liczbę użytkowników z uwzględnieniem jej zmiany oraz intensywności wykorzystywania przez nich zasobów dyskowych.

Pozostały jeszcze do ustalenia wartości limitów czasowych. Służy do tego również polecenie *edquota*. Uruchomione z opcją *-t* ustala wartości obowiązujące w danym systemie plików. Działanie komendy polega na uruchomieniu edytora, do którego załadowany zostaje plik tymczasowy utworzony na podstawie zawartości baz danych systemu kontyngentów. Po jego edycji uruchamiany jest parser, który po udanym odczytaniu wpisanych danych zapisuje je do bazy danych systemu kontyngentów. Wyjście z edytora bez zapisu lub błędny format pliku powodują, że żadne dane nie zostaną zapisane w bazie danych. Przykład ustawienia wartości limitów czasowych zamieszczono poniżej:

```

1 Time units may be: days, hours, minutes, or seconds
2 Grace period before enforcing soft limits for users:
3 /home: block grace period: 5 days, file grace period: 7 days
4 /tmp: block grace period: 5 days, file grace period: 7 days

```

Jak widać zastosowano najczęściej wykorzystywane wartości limitów czasowych. Dla bloków dyskowych jest to 5 dni, dla i-węzłów 7. Wartości można podawać w dniach, godzinach, minutach i sekundach. Ustalone limity dla jednego użytkownika mogą być kopiowane innym. Komenda *edquota* wymaga wówczas użycia opcji *-p* oraz podania dwóch argumentów. Pierwszym jest nazwa lub numer identyfikacyjny użytkownika, którego ustawienia stanowiąc będą wzorzec. Drugim, nazwa lub numer identyfikacyjny użytkownika, któremu wartości te mają zostać nadane.

Raport o wykorzystaniu limitowanych zasobów systemów plików w systemach BSD dostarcza polecenie *repquota*. Argumentem wywołania może być punkt montowania interesującego nas systemu plików. Pominięcie argumentu jest możliwe jeśli użyta zostanie opcja **-a**. W tym przypadku wypisana zostanie informacja o wykorzystaniu zasobów we wszystkich systemach plików zamontowanych z opcjami wymuszającymi działanie systemu kontyngentów. Dla uzyskania informacji o tym, którego systemu plików dany fragment raportu dotyczy, zaleca się użycie opcji **-v**. Przykładowy wynik działania komendy przedstawiono poniżej.

```

1  -su-3.2# repquota -av
2  *** Report for group quotas on /home (/dev/ad0s1e)
3
4          Block limits
5  Group      used  soft  hard  grace  used  soft  hard  grace
6  wheel      --    2    0    0    -    1    0    0    -
7  operator   --    2    0    0    -    1    0    0    -
8  antek      --   18    0    0    -    9    0    0    -
9  jan        -- 18064    0    0    -   69    0    0    -
10
11 *** Report for user quotas on /home (/dev/ad0s1e)
12
13          Block limits
14  User      used  soft  hard  grace  used  soft  hard  grace
15  root      --    4    0    0    -    2    0    0    -
16  antek     --   18    0    0    -    9    0    0    -
17  jan       -- 18064 25000 30000 -   69 5000 8000 -
18
19 *** Report for group quotas on /tmp (/dev/ad0s1f)
20
21          Block limits
22  Group      used  soft  hard  grace  used  soft  hard  grace
23  wheel      -- 37090    0    0    - 1249    0    0    -
24  operator   --    2    0    0    -    1    0    0    -
25
26 *** Report for user quotas on /tmp (/dev/ad0s1f)
27
28          Block limits
29  User      used  soft  hard  grace  used  soft  hard  grace
30  root      -- 22644    0    0    - 1229    0    0    -
31  jan       +- 14448 8000 10000 7    21 3000 6000 -

```

Raport ma budowę zwrotkową. Zwrotka opisuje wykorzystanie zasobów w danym systemie plików, identyfikowanym przez punkt montowania oraz nazwą pliku urządzenia, na którym system ten został założony. Informacja jest podzielona na część opisującą grupy oraz użytkowników. Znaczenie kolumn jest identyczne jak w raporcie dostarczonym przez komendę *quota*, za wyjątkiem kolumny drugiej. Zawiera ona symboliczne oznaczenie wykorzystania zasobów: znak **-** oznacza, że ograniczenie miękkie nie zostało przekroczone, zaś znak **+**, że użytkownik lub grupa wykorzystują większą ilość zasobów niż ograniczenie to wynosi. Raport jest generowany dla użytkowników lub grup, które we wskazanych systemach plików wykorzystują co najmniej jedną jednostkę reglamentowanego zasobu.

Przekroczenie ograniczenia na zasoby jest sygnalizowane w raporcie polecenia *quota* znakiem **\*** za liczbą określającą wykorzystanie danego zasobu. W przypadku użytkownika *jan* wygląda to następująco:

```

1  -bash-3.2$ quota -v
2  Disk quotas for user jan (uid 1002):
3      Filesystem  usage   quota   limit   grace   files   quota   limit   grace
4      /home      18064   25000   30000           69   5000   8000
5      /tmp      14448*  8000   10000         7    21   3000   6000

```

W systemie plików */home* użytkownik *jan* może jeszcze wykorzystać 11926 bloków danych i 7931 i-węzłów. W systemie plików */tmp* zostało przekroczone ograniczenie twarde. Jak to możliwe? Ograniczenie na zasoby zostało wprowadzone dla użytkownika po tym, jak utworzył on w systemie pliki i to na poziomie niższym niż wykorzystywane przez nie zasoby. W takiej sytuacji użytkownik może tworzyć jedynie puste pliki. Możliwość wykorzystania bloków danych pojawi się dopiero po ograniczeniu poziomu ich wykorzystania poniżej wartości ograniczenia twardego i to jedynie przez okres 7-dni.

W analogiczny sposób przebiega proces ustalania poziomów wykorzystania zasobów oraz limitów czasowych dla grup użytkowników. Dla przykładu, do systemu zostali dodani dwaj użytkownicy: *wacek* i *wojtek*. Ich grupą podstawową jest grupa *pracownicy*. Stąd tworzone przez nich pliki będą miały jako właściciela grupowego tę właśnie grupę. Rozpoczynamy od zdefiniowania ograniczeń na bloki danych oraz i-węzły. W tym celu uruchamiamy polecenia *edquota* z opcją *-g* i wartością będącą nazwą lub numerem identyfikacyjnym grupy w systemie:

```

1  -su-3.2# edquota -g pracownicy

```

W wyniku jej działania, w katalogu */tmp* zostaje utworzony tekstowy plik tymczasowy zawierający wartości ograniczeń dla wskazanej grupy we wszystkich systemach plików zamontowanych z opcją *groupquota*. Uruchomiony zostanie edytor z utworzonym plikiem tymczasowym zawierającym aktualne wartości ograniczeń. Następnie należy wprowadzić ustalone poziomy limitów dla systemów plików, zapisać plik i wyjść z edytora. Jak wspomniano wcześniej, zmiana zawartości pliku oraz jego zapis wymusi uruchomienie programu, który sprawdzi poprawność składniową pliku. Jeśli nie stwierdzi w nim błędów, nowe wartości zostaną zapisane w plikach bazy danych systemu kontyngentów. W przypadku przeciwnym zostaniemy poinformowani o błędzie, a zmiany nie zostaną zapisane w plikach bazy danych. Przykładowe ustawienia dla grupy *pracownicy* przedstawiono poniżej:

```

1  Quotas for group pracownicy:
2  /home: kbytes in use: 0, limits (soft = 500000, hard = 550000)
3         inodes in use: 0, limits (soft = 150000, hard = 180000)
4  /tmp: kbytes in use: 0, limits (soft = 250000, hard = 280000)
5         inodes in use: 0, limits (soft = 100000, hard = 120000)

```

Ostatnią czynnością jest ustalenie limitów czasowych. W tym celu uruchamiamy komendę *edquota* z opcjami *-t* oraz *-g*. W systemie istnieje możliwość wprowadzenia wartości limitów czasowych obowiązujących wszystkie grupy. Dlatego nie podajemy nazwy ani numeru grupy. W tym przypadku do edytora zostanie wczytany plik zawierający obowiązujące wartości. Nowe wartości ustalamy na najczęściej wykorzystywane 7-dnio dniowe, a następnie zapisujemy plik i kończymy pracę edytora:

```

1  Time units may be: days, hours, minutes, or seconds
2  Grace period before enforcing soft limits for groups:

```

```

3 /home: block grace period: 7 days, file grace period: 7 days
4 /tmp: block grace period: 7 days, file grace period: 7 days

```

Dla uzyskania informacji o wykorzystaniu zasobów przez grupę użytkowników komendę *quota* należy uruchomić z opcjami **-v** oraz **-g**. W przypadku zwykłego użytkownika informacja będzie dotyczyć jedynie jego grupy podstawowej i nie ma on możliwości uzyskania informacji o innej grupie, nawet jeśli do niej należy. Użytkownik *root* uzyska informację o wykorzystaniu zasobów przez wszystkie grupy, do których należy. Może on także uzyskać informacje dotyczące dowolnej grupy, podając jej nazwę lub numer identyfikacyjny jako argument wywołania komendy. Przykładowy wynik przedstawiono poniżej:

```

1 -su-3.2$ quota -vg pracownicy
2 Disk quotas for group pracownicy (gid 1003):
3   Filesystem  usage  quota  limit  grace  files  quota  limit  grace
4   /home      45792 500000 550000          1 150000 180000
5   /tmp       2724 250000 280000          26 100000 120000

```

**Składnia i podstawowe opcje poleceń systemu** Opcje użyte w poleceniach wykorzystywanych w powyższych przykładach są najczęściej wykorzystywanymi podczas administrowania systemem kontyngentów. Niemniej jednak w szczególnych przypadkach konieczne staje się użycie poleceń obsługujących system kontyngentów z mniej popularnymi opcjami. Poniżej przedstawiono ich krótki opis.

Polecenie *quotacheck* służy do obliczenia wykorzystania zasobów w systemach plików i zapisania wyniku w plikach bazy danych systemu kontyngentów. Użycie polecenia bez argumentu spowoduje, że komenda rozpatrzy wszystkie systemy plików, zdefiniowane w pliku */etc/fstab*, które zostały zamontowane z opcjami umożliwiającymi działanie systemu kontyngentów. Argumentem może być punkt montowania wybranego systemu plików. Wówczas działanie polecenia ograniczy się jedynie do tego systemu plików, ale on również musi być zamontowany ze stosownymi opcjami. W dystrybucji FreeBSD polecenie *quotacheck* posiada następujące opcje:

- **-a** – uruchomienie komendy z tą opcją spowoduje weryfikację użycia zasobów przez użytkowników indywidualnych i grupy w systemach plików, które w pliku */etc/fstab* posiadają zdefiniowane opcje montowania umożliwiające działanie systemu kontyngentów.
- **-g** – opcja umożliwia weryfikację jedynie ograniczeń nałożonych na grupy użytkowników.
- **-l *maxrun*** – opcja umożliwia określenie maksymalnej liczby jednocześnie weryfikowanych systemów plików. Wartość domyślna wynosi 0, co oznacza weryfikację sekwencyjną.
- **-u** – użycie opcji powoduje sprawdzenie limitów nałożonych na użytkowników indywidualnych.
- **-v** – opcja umożliwia wypisanie różnic między aktualnie wyznaczonymi, zapisanymi w plikach bazy danych wartościami wykorzystania zasobów systemów plików. Przykładowy raport zamieszczono poniżej:

```

1 *** Checking user and group quotas for /dev/ad0s1e (/home)
2 *** Checking user and group quotas for /dev/ad0s1f (/tmp)
3 /tmp: root      fixed (user):    inodes 1365 -> 1377      blocks 59380 -> 59428
4 /tmp: wheel     fixed (group):    inodes 1230 -> 1242      blocks 136872 -> 136920

```

Komenda *quotaon* służy do uczynienia systemu kontyngentów aktywnym, czyli wymuszającym na użytkownikach i grupach zdefiniowane ograniczenia. Wymagane jest, aby w systemie plików, dla którego uaktywnia się system kontyngentów, w jego katalogu głównym istniały pliki bazy danych. W przypadku dystrybucji FreeBSD są to pliki *quota.user* oraz *quota.group*, przechowujące informacje o wykorzystywanych zasobach odpowiednio przez użytkowników oraz grupy. Polecenie *quotaon* jako argument wymaga podania punktu montowania systemu plików, dla którego system kontyngentów ma zostać aktywowany. Argument może zostać pominięty, ale wymaga to użycia opcji **-a**. Spowoduje to aktywowanie systemu kontyngentów w każdym systemie plików, który został zamontowany z opcjami uruchamiającymi system kontyngentów. Wyjątek stanowią sieciowe systemy plików.

Do najczęściej wykorzystywanych opcji komendy *quotaon* należy zaliczyć:

- **-a** – opcja powoduje uaktywnienie systemu kontyngentów w stosunku do użytkowników i grup, w każdym systemie plików, który został zamontowany z opcjami umożliwiającymi odczyt, zapis oraz działanie systemu kontyngentów. Domyślnie zostaną uaktywnione jedynie funkcjonalności systemu kontyngentów zdefiniowane w pliku */etc/fstab*.
- **-g** – użycie opcji spowoduje aktywację systemów kontyngentów dla grup użytkowników, w odniesieniu do systemów plików dla których opcje montowania *groupquota* zostały wyspecyfikowane w pliku */etc/fstab*.
- **-u** – opcja umożliwia aktywację systemów kontyngentów dla użytkowników, w odniesieniu do systemów plików dla których opcje montowania *userquota* zostały wyspecyfikowane w pliku */etc/fstab*.
- **-v** – opcja włącza gadatliwy tryb raportowania komendy, który powoduje wypisanie komunikatu o każdym systemie plików, dla którego system kontyngentów został włączony.

Polecenie *quotaoff* jest symetrycznym do polecenia *quotaon* i służy do uczynienia systemu kontyngentów nieaktywnym. Argumentem wywołania jest punkt montowania systemu plików. Argument wywołania może zostać pominięty, ale wymaga to użycia opcji **-a**. W takim wypadku analizowana jest zawartość pliku */etc/fstab* i system quota jest wyłączany dla wszystkich systemów plików, których zdefiniowane w tym pliku opcje montowania umożliwiają jego działanie w odniesieniu do użytkowników lub grup. Jak wspomniano, najczęstszym powodem wyłączania systemu kontyngentów jest konieczność uruchomienia polecenia *quotacheck* w celu uaktualnienia zawartości bazy danych o wykorzystywanych zasobach systemów plików.

W systemie FreeBSD opcje polecenia *quotaoff* są identyczne, jak opcje polecenia *quotaon*.

Polecenie *edquota* to edytor ograniczeń i limitów czasowych dla systemu kontyngentów. Domyślnie argumentem wywołania może być nazwa lub numer identyfikacyjny jednego lub kilku użytkowników, dla których definiowane są ograniczenia. Wywołanie takie umożliwia również użycie opcji **-u**. Jeśli ograniczenia mają zostać ustawione dla jednej lub kilku grup, to wymagane jest użycie opcji **-g** oraz podanie jako argumentu ich nazw lub numerów identyfikacyjnych. Jak wspomniano, edycja ustawień odbywa się w oparciu o tymczasowy plik tekstowy, utworzony na podstawie zawartości bazy danych systemu kontyngentów w katalogu */tmp*, a edytowanego edytorem zdefiniowaną wartością zmiennej środowiskowej *EDITOR*. Jeśli w pliku tymczasowym dokonano zmian, to zakończenie pracy edytora uruchamia program sprawdzający poprawność jego zawartości. W przypadku wykrycia błędu, na ekranie pojawia się stosowny komunikat, a zawartość bazy danych nie jest modyfikowana. W przeciwnym wypadku nowe ustawienia zostają zapisane w bazach danych systemu kontyngentów.

Polecenie *edquota* posiada następujące opcje:

- **-e** – powoduje, iż polecenie *edquota* pracuje w trybie interaktywnym. Wartości ograniczeń podaje się wówczas w linii komend. Obowiązuje następująca składnia: **-e fspath[:bslim[:bhlm[:islim[:ihlim]]]]**. Wymagane jest zatem podanie punktu montowania systemu plików. Ograniczenia zasobów podaje się w kolejności: miękkie i twarde ograniczenie na liczbę bloków danych oraz miękkie i twarde ograniczenia na liczbę i-węzłów. W jednym wywołaniu możliwe jest ustalenie ograniczeń obowiązujących w wielu systemach plików, ale nałożonych na jednego użytkownika lub grupę użytkowników. Wymaga to powtórzeń opcji **-e**. Przykładowo:

```

1 -su-3.2# edquota -e /home:100000:125000:2000:2500 -e /tmp:60000:75000:4000:6000 jan
2 -su-3.2# quota -v jan
3 Disk quotas for user jan (uid 1002):
4
5      Filesystem      usage    quota    limit    grace    files    quota    limit    grace
6      /home           18064   100000   125000
7      /tmp             45792   60000    75000

```

- **-f** – umożliwia wyspecyfikowanie ograniczeń dla systemu plików wskazanego wartością opcji. Może nią być punkt montowania lub nazwa pliku urządzenia, na którym dany system plików został założony. Opcja ta najczęściej stosowana jest z opcją **-p**, gdzie ogranicza kopiowanie ustawień wzorcowych jedynie do wskazanego systemu plików.
- **-g** – domyślnie polecenie *edquota* umożliwia edycję ograniczeń dla użytkowników indywidualnych. Opcja **-g** umożliwia zarządzanie ograniczeniami dla grup użytkowników. Opcja ta jest stosowana z opcją **-p** w celu kopiowania ustawień grupy wzorcowej dla innych grup użytkowników lub z opcją **-t** w celu ustalenia wskazanej grupie indywidualnych limitów czasowych dla miękkich ograniczeń. Wartością opcji może być nazwa lub numer identyfikacyjny grupy.
- **-p** – opcja służy do kopiowania ustawień zdefiniowanych dla jednego użytkownika lub grupy odpowiednio innym użytkownikom i grupom. Argumentem opcji może być nazwa lub numer identyfikacyjny użytkownika lub grupy, których ustawienia zostaną potraktowane jako wzorcowe.
- **-u** – opcja najczęściej stosowana z opcją **-t**. Umożliwia wówczas indywidualne ustalanie limitów czasowych dla użytkownika, którego nazwa lub numer identyfikacyjny pojawiły się jako wartość opcji, we wskazanym systemie plików.
- **-t** – opcja umożliwia ustalenie limitów czasowych dla miękkich ograniczeń.

Polecenie *quota* dostarcza informacji o wykorzystaniu zasobów systemów plików, w których system kontyngentów jest aktywny oraz wartościach zdefiniowanych ograniczeń i limitów czasowych. Domyślnie informacja dotyczy użytkownika. Zwykły użytkownik może uzyskać informacje dotyczące jedynie jego jako użytkownika indywidualnego oraz swojej grupy podstawowej, jeśli na nią zostały nałożone ograniczenia. Użytkownik *root* może uzyskać informację o każdym użytkowniku i każdej grupie zdefiniowanych w systemie. Rozmiar bloku, w którym podawane jest zużycie oraz ograniczenia dla bloków dyskowych wynosi 1024 B. Podstawowe opcje polecenia to:

- **-f ścieżka\_dostępu** – wartością opcji może być ścieżka dostępu do pliku lub katalogu znajdującego się wewnątrz zamontowanego systemu plików. Jej użycie spowoduje, że wypisana przez komendę *quota* informacja będzie dotyczyć jedynie tego systemu plików.

- **-g** – opcja umożliwia wypisanie informacji o wykorzystanych zasobach, ograniczeniach i limitach dla grup, których użytkownik jest członkiem.
- **-h** – opcja powoduje wypisanie informacji w postaci czytelnej, z wykorzystaniem jednostek: *(B)yte*, *(K)ilobyte*, *(M)egabyte*, *(G)igabyte*, *(T)erabyte* oraz *(P)etabyte*.
- **-l** – użycie opcji spowoduje, że nie pojawi się informacja o sieciowych systemach plików.
- **-q** – opcja umożliwia wypisanie informacji w postaci zwężonej, ograniczonej jedynie do systemów plików, w których ograniczenia miękkie zostały przekroczone.
- **-r** – to opcja umożliwiająca formatowanie wypisanej informacji do postaci zwrotki zawierającej informację o wykorzystanych zasobach, ograniczeniach i limitach dla jednego systemu plików. Dodatkowo, format pól zawierających informacje o czasie jest taki, jak zwraca funkcja *ctime*. Opcja ta implikuje użycie opcji **-v**. Przykład zamieszczono poniżej:

```

1  -su-3.2$ quota -r
2  Raw user quota information for id 1002 on /home
3  block hard limit:      250000
4  block soft limit:     200000
5  current block count:   36128
6  i-node hard limit:     2500
7  i-node soft limit:     2000
8  current i-node count:  69
9  block grace time:      1285320245 Fri Sep 24 11:24:05 2010
10 i-node grace time:     1285320245 Fri Sep 24 11:24:05 2010

```

- **-u** – opcja domyślna, służy wypisaniu informacji dotyczącej użytkownika. Użycie z opcją **-g** umożliwia wypisanie informacji dotyczącej użytkownika oraz grup.
- **-v** – opcja umożliwia wypisanie informacji również o systemach plików, w których aktywny jest system kontyngentów, ale nie ustawiono ograniczeń i limitów czasowych.

Polecenie *repquota* dostarcza informacji o wykorzystaniu zasobów oraz nadanych wartościach ograniczeń i limitów czasowych w systemach plików. Argumentem może być punkt montowania lub nazwa pliku urządzenia, na którym dany system plików się znajduje, co najmniej jednego systemu plików. Argument można pominąć, ale wymaga to użycia opcji **-a**. Polecenie efektywnie działa, jeśli uruchomi je użytkownik *root*. Zwykły użytkownik otrzymuje informację o braku dostępu do plików bazy danych systemu kontyngentów. Do podstawowych opcji polecenia *repquota* należy zaliczyć:

- **-a** – opcja umożliwia wypisanie raportu o stanie systemu kontyngentów dla wszystkich systemów plików zdefiniowanych w pliku */etc/fstab* z odpowiednimi opcjami montowania.
- **-g** – użycie opcji powoduje wypisanie raportu dotyczącego grup użytkowników. Domyślnie wypisywana jest informacja dotycząca użytkowników indywidualnych i grup.
- **-n** – opcja umożliwia wypisanie numerów identyfikacyjnych użytkowników i grup zamiast ich nazw.
- **-u** – opcja ogranicza wypisywaną informację do użytkowników indywidualnych.



- **-v** – to opcja formatująca wypisywaną informację. Jej użycie powoduje wypisanie nagłówka przed informacjami dotyczącymi kolejnych systemów plików.

W dystrybucji FreeBSD nie został zaimplementowany, dostępny w dystrybucjach RedHat, program narzędziowy *warnquota*. Przypomnijmy, że służy on do powiadamiania użytkowników przez pocztę elektroniczną o przekroczeniu ograniczeń nałożonych na wykorzystanie zasobów systemów plików i wyczerpaniu się limitów czasowych. Program ten najczęściej jest uruchamiany przez demon zegarowy *cron*. Prosty jego substytut można zaimplementować wykorzystując raporty generowane przez polecenie *repquota*.

### 3.3.5 Dziennikowane systemy plików

Dziennik jest podstawowym elementem wyróżniającym systemy plików tego rodzaju. Służy on przechowywaniu danych o zmianach informacji przechowywanej w systemie plików. Dane te są rejestrowane w dzienniku jeszcze przed ich fizyczną realizacją. Systemy plików pozbawione kroniki muszą, w celu zagwarantowania spójności zmian, stosować synchroniczne operacje zapisu. Zwiększa to liczbę ruchów głowicą zmniejszając tym samym efektywność systemu plików. Stąd praktycznie każdy system plików nie wyposażony w mechanizm dziennikowania korzysta z pewnych mechanizmów kompletowania operacji zapisu, których wynikiem jest minimalizacja liczby ruchów głowic oraz wynikające stąd skrócenie czasu, w którym dostęp do systemu plików jest blokowany w wyniku realizowania synchronicznej operacji zapisu.

Kolejnym istotnym aspektem, na który należy zwrócić uwagę, jest stosowanie mechanizmu buforów dyskowych. Powstał on w celu opóźniania instrukcji zapisu oraz przyspieszania odczytu z dysku. Powoduje on jednak, że podczas normalnej pracy systemu operacyjnego, w buforach dyskowych niemal zawsze znajduje się pewna liczba oczekujących na realizację operacji zapisu. Jak wiemy podczas operacji odmontowania systemu plików, realizowanej przykładowo podczas każdego zamykania systemu operacyjnego, jądro systemu operacyjnego zapewnia opróżnienie zawartości buforów dyskowych. Jednak starsze wersje systemów nie zawsze o to dbały. Dlatego niejako na wyrost (kilkakrotnie przed zamknięciem systemu operacyjnego) używało się w nich komendy *sync*, traktując ją jako środek na zapewnienie spójności systemów plików.

Przypomnijmy, że podczas montowania systemu plików, jądro systemu operacyjnego dokonuje aktualizacji listy wolnych i-węzłów. W przypadku systemów plików nie wykorzystujących mechanizmu dziennikowania nie jest natomiast w stanie zweryfikować zawartości bloków danych i listy wolnych bloków danych. Może to wynikać z faktu, że zmiany wprowadzone do systemu plików na skutek awarii zostały odnotowane w nim jedynie częściowo i brak jest możliwości określenia ich aktualnego stanu. Weryfikacja taka po prawidłowym zamknięciu systemu plików jest niepotrzebna, ponieważ procedura odmontowania zawsze oczekuje na zakończenie zaplanowanych operacji zapisu i zaznacza w bloku identyfikacyjnym, że dany system plików jest poprawny. Dla tak oznaczonego systemu plików, podczas jego montowania nie są wywoływane programy narzędziowe w celu jego weryfikacji.

Spójność dziennikowanego systemu plików możemy przywrócić odtwarzając te z operacji odnotowanych w dzienniku, które nie są oznaczone jako zakończone, a zatem utrwalone na dysku. Do operacji tych należy m.in. zapis metadanych oraz zapis wszystkich zmian danych przechowywanych w dowolnym pliku. Należy jednak uświadomić sobie fakt, iż nie zawsze możliwe jest przywrócenie wszystkich zmian dokonanych w systemie plików z kroniką, gdyż informacje o nich mogły zostać zapisane w dzienniku w sposób niepełny. Uwzględniając zastosowane rozwiązania implementacyjne można założyć, że w większości przypadków system plików z kroniką może zostać przywrócony do stanu spójnego jedynie na podstawie zawartości dziennika bez konieczności wyczerpującej analizy zawartości całego systemu plików.

Dziennikowane systemy plików wykorzystują dwa rodzaje zapisu dziennika. Dziennik może zawierać:

1. Zapis zmian związanych z wykonaniem każdej operacji zapisu wprowadzonych jedynie do metadanych systemu plików.
2. Zapis zmian związanych z wykonaniem każdej operacji zapisu wprowadzonych zarówno do danych przechowywanych w plikach jak również do metadanych systemu plików.

W obu rozwiązaniach spójność systemu plików zostanie przywrócona dzięki rejestracji zmian metadanych. W drugim przypadku, bardziej żmudnym i czasochłonnym, istnieje możliwość odzyskania co najmniej części zapisywanych danych. Wpisy dokonywane w dzienniku mają charakter tranzakcyjny, co w praktyce oznacza, że rejestrowany jest wpis zawierający cały zestaw zmian koniecznych do przeprowadzenia, aby dana operacja została wykonana jako kompletna. Klasycznym przykładem jest zapisanie nowej wersji pliku. Operacja ta wymaga:

- Znalezienia nowych bloków danych, w których będzie przechowywana informacja dopisana na końcu pliku.
- Zaznaczenia znalezionych bloków danych jako zajętych.
- Zapisania do nowo znalezionych bloków danych zmodyfikowanego pliku danych.
- Modyfikacji informacji o zajmowanych przez plik blokach danych przechowywana w opisyującym go i-węźle.
- Aktualizacji informacji o czasie ostatniego dostępu i modyfikacji pliku przechowywana również w opisyującym go i-węźle.

Realizacja transakcji, w takim przypadku, będzie polegała na zapisaniu w pliku dziennika wszystkich czynności koniecznych do wykonania, aby transakcja ta została uznana jako kompletna. Następnie, po wykonaniu każdej czynności, informacja o niej zostanie usunięta z dziennika. Jeśli w tym momencie nastąpi awaria systemu, to po jego ponownym uruchomieniu, podczas montowania systemu plików, w którym przebiegała przykładowa transakcja, jego dziennik będzie zawierał jedynie informacje o czynnościach, które należy wykonać aby operacja ta została skompletowana. W niektórych systemach plików stosowane jest podejście odwrotne, polegające na odnotowywaniu w dzienniku czynności już wykonanych. Przy takim podejściu, podczas montowania systemu plików po awarii systemu dokonujemy analizy dziennika i na podstawie uzyskanych z niej informacji wykonywane są jedynie te czynności, które są konieczne aby skompletować rozpoczęte transakcje. Czynności związane z każdą zmianą zawartości systemu plików nazywane są *niepodzielnymi* lub *atomowymi*, gdyż w praktyce konieczne jest ich wykonanie w całości lub wycofanie.

Rozpatrzmy przedstawiony przykład w kontekście mechanizmów dziennikowania rajetsrujących jedynie zmiany metadanych systemu plików. System taki zarejestrowałby wszystkie przytoczone zmiany, ale nie zarejestrowałby zawartości nowych bloków danych. Odtworzenie niepodzielnych operacji na metadanych zagwarantuje spójność systemu plików. Po jego montowaniu nie będzie konieczna analiza całego systemu plików w celu wyznaczenia nowej listy wolnych i-węzłów, a jedynie korekta istniejącej w oparciu o zawartość dziennika. Jednak w pliku mogą pojawić się „śmieci”, gdyż dziennik nie zawiera informacji o danych wprowadzonych do dołączonych do pliku blokach danych. W takim schemacie dziennikowania, dziennik powinien również rejestrować stan systemu plików z momentu poprzedzającego wprowadzenie zmian. Na ich podstawie możliwe byłoby zdefiniowanie zakresu operacji wycofania (ang. *undo*), która w tym przypadku polegałaby

na doprowadzeniu systemu plików do stanu sprzed wykonania operacji na jego metadanych, a zatem stanu sprzed rozpoczęcia wykonywania niezakończonych transakcji. Rejestrując zmiany metadanych oraz dane, które w systemie plików ulegną modyfikacji mamy gwarancję, że system plików po ewentualnej awarii pozostanie spójny oraz, że w przechowywanych w nim plikach nie pojawią się dane przypadkowe.

Należy sobie jednak zdawać sprawę, że im więcej informacji jest rejestrowanych w dzienniku, tym więcej czasu potrzeba na utrwalenie zapisów, zwłaszcza że zapis w dzienniku musi odbywać się synchronicznie. Z drugiej strony przechowywanie większej ilości danych w dzienniku pozwala na odzyskanie jego obrazu, który jest bliższy temu sprzed awarii. Z punktu widzenia użytkownika systemu stanowi to istotną redukcję ryzyka utracenia wprowadzonych modyfikacji przed awarią systemu. Praktycznie każdy system plików stosuje inne mechanizmy dziennikowania, optymalne z punktu widzenia zastosowanej w nim organizacji danych na dysku.

Ostatnią kwestią, mającą istotny wpływ na efektywność systemu plików z mechanizmem dziennikowania, jest położenie dziennika. Rozwiązanie najprostsze z punktu widzenia implementacji polega na przechowywaniu dziennika w postaci pliku regulanego wewnątrz systemu plików, który podlega procesowi rejestracji zmian. Podejście takie rodzi jednak dwa oczywiste problemy. Po pierwsze, zapis do pliku dziennika musi odbywać się z wykorzystaniem funkcji danego systemu operacyjnego, stąd utrzymanie aktualności dziennika może negatywnie wpływać na wydajność systemu plików. Po drugie, uszkodzenie systemu plików, w którym przechowywany jest dziennik, może powodować również utratę zawartości dziennika, co w większości przypadków pozbawi nas szans na odzyskanie ostatnio zapisanej w nim informacji.

Drugim miejscem, w którym można przechowywać dziennik systemu plików jest specjalny obszar systemu plików, niedostępny dla programów użytkowych. W tym rozwiązaniu, operacje wykonywane na pliku dziennika mogą zostać zrealizowane z wykorzystaniem specjalnych funkcji, zoptymalizowanych pod kątem wydajności zapisu. Przy takim podejściu zmniejszeniu ulega również ryzyko utraty dziennika w wyniku uszkodzenia systemu plików. W prawdzie dziennik znajduje się nadal w systemie plików, ale w jego specjalnym obszarze, który nie jest dostępny dla standardowych operacji zapisu. W stosunku do rozwiązania omówionego poprzednio, pozostał problem rywalizacji operacji zapisu do dziennika z operacjami zapisu do systemu plików.

Stąd ostatnią możliwą lokalizacją dziennika jest obszar poza systemem plików, przydzielony w specjalnym obszarze dysku. Rozwiązanie to eliminuje problem wpływu uszkodzeń systemu plików na zawartość jego dziennika, jak również rywalizacji operacji zapisu pod warunkiem, że system plików i jego dziennik są przechowywane na różnych fizycznych dyskach. Dochodzimy do ważnej przesłanki wpływającej na wydajność systemu plików wyposażonego w mechanizm dziennikowania, mówiącej, że w celu uzyskania wzrostu wydajności i poziomu bezpieczeństwa, system plików i jego dziennik należy, o ile to możliwe, umieścić na różnych dyskach.

Podsumowując:

- Dziennikowaniu mogą podlegać jedynie zmiany metadanych systemu plików oraz zmiany metadanych i danych przechowywanych w plikach.
- Mechanizm dziennikowania może polegać na zapisaniu w pliku dziennika całej transakcji, a następnie usuwaniu z niego informacji o czynnościach już wykonanych. W takim podejściu, po awarii systemu, w dzienniku systemu pozostaje informacja o czynnościach, które mają zostać wykonane w celu skompletowania transakcji. Podejście alternatywne polega na zapisywaniu czynności już wykonanych. Po awarii systemu w dzienniku mamy dostępną informację o stanie zaawansowania transakcji, która pozwala nam na jej skompletowanie.
- Dziennik systemu plików może zostać zrealizowany na kilka sposobów. Rozwiązaniem optymalnym z punktu widzenia wydajności systemu plików oraz bezpieczeństwa przechowywa-

nych w nim danych jest umieszczenie systemu plików i jego dziennika na różnych fizycznych dyskach.

- Podczas wykonywania procedury montowania systemu plików z mechanizmem dziennikowania przeglądany jest jego dziennik w celu znalezienia w nim transakcji zatwierdzonych do wykonania, a nie zaznaczonych jako zakończone. Na podstawie zawartości dziennika dokonywane jest więc uspoźnienie zawartości systemu plików. W przypadku systemu plików bez mechanizmu dziennikowania możliwe jest jedynie uaktualnienie listy wolnych i-węzłów, ale po analizie zawartości całego systemu plików. Dołączanie do nich bloków danych byłoby ryzykowne, gdyż przechowywana w nich informacja, z powodu braku mechanizmu dziennikowania, będzie przypadkowa.

### 3.4 Administracja wybranymi systemami plików w dystrybucji RedHat

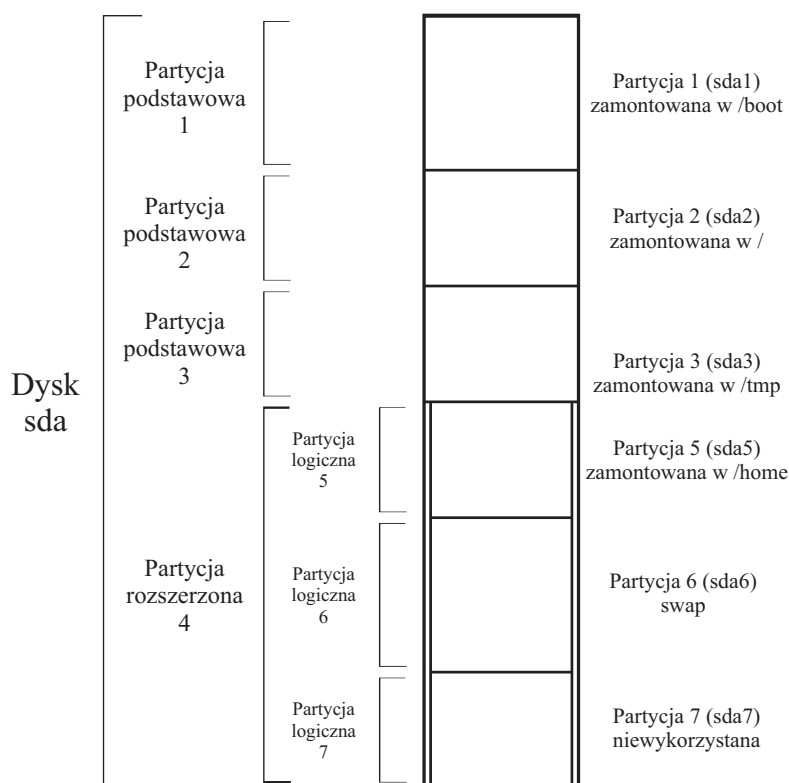
Praktycznie każda komercyjna dystrybucja systemu uniksowego stosuje własny system plików, promując go jako charakteryzujący się największą wydajnością przy zachowaniu wysokiego poziomu bezpieczeństwa przechowywanych danych. Należy jednak zwrócić uwagę, iż wydajność systemu plików silnie zależy od charakteru przechowywanych w nim danych. W szczególności różne formaty plików charakteryzują się specyfiką operacji dostępu, a ich wydajność zależy od sposobu przechowywania informacji na dysku. Nie można zatem stwierdzić, iż istnieje uniwersalny w zastosowaniu system plików. Należy być świadomym, że rodzaj systemu plików należy dobierać pod kątem charakteru danych, które w nim będą przechowywane. Stąd intencją tego podrozdziału jest omówienie kilku najpopularniejszych systemów plików oraz wskazanie zastosowań, w których ich zalety zostaną w pełni wykorzystane.

#### 3.4.1 Nazwy plików urządzeń w dystrybucji RedHat

Przykładową organizację dysku w systemie rodziny RedHat przedstawiono na rysunku 3.11.

Jak widać hierarchia składa się z dwóch stopni, a mianowicie dysku fizycznego i występujących na nim partycji. Nazwa pliku reprezentującego urządzenie składa się z trzech części:

1. Pierwsza określa typ urządzenia dyskowego. Oznaczenia najczęściej występujących przedstawiono poniżej:
  - **hd** – nazwa stosowana w odniesieniu do dysków z interfejsem IDE. Kolejna litera oznaczała miejsce podłączenia: **a** – master na pierwszej tasiemce interfejsu IDE, **b** – slave na pierwszej tasiemce interfejsu IDE, **c** oraz **d** odpowiednio master i slave na drugiej tasiemce. Od dystrybucji Fedora 8 oznaczenie to nie jest używane.
  - **sd** – dyski z interfejsem SCSI. Od dystrybucji Fedora 9 wszystkie dyski twarde, niezależnie od ich interfejsu są „widziane” jako SCSI. Numerowane są literami, od litery **a** począwszy.
  - **scd** – napędy SCSI CD-ROM.
  - **fd** – napęd dyskietek elastycznych.
  - **st** – napęd tasiemki wykorzystujący interfejs SCSI.
  - **nst** – napęd tasiemki SCSI bez możliwości przewijania.
  - **ht** – napęd tasiemki SCSI.



Rysunek 3.11: Organizacja dysku w systemie operacyjnym dystrybucji RedHat.

- **md** – urządzenia RAID.
  - **cdrom** – dowiązanie symboliczne do pliku reprezentującego napęd CD-ROM, zdefiniowane w pliku `/etc/udev/rules.d`.
  - **cdrecorder** – dowiązanie symboliczne do pliku reprezentującego napęd CR-R lub CD-RW.
  - **floppy** – dowiązanie symboliczne do pliku reprezentującego napęd dyskietek elastycznych.
  - **tape** – dowiązanie do pliku reprezentującego napęd taśmowy.
2. Druga to litera będąca numerem urządzenia danego typu zainstalowanego w systemie. Numeracja rozpoczyna się od litery a.
  3. Trzecia określa numer partycji na dysku. Numeracja rozpoczyna się od 1. Na dysku możemy utworzyć cztery partycje. Wyróżniamy partycje typu podstawowego (*primary*) oraz typu rozszerzonego (*extended*). Dodatkowe ograniczenie dotyczy możliwości występowania co najwyżej jednej partycji typu rozszerzonego na danym dysku. Partycję tego typu możemy jednak dzielić na partycje logiczne omijając w ten sposób limit czterech partycji. Niestety partycje logiczne dla niektórych systemów operacyjnych posiadają „mniejszą wartość” np. ze względu na brak możliwości inicjalizacji z ich wykorzystaniem.

Przykładowo plik `/dev/sda3` reprezentuje partycję, której nadano numer 3, znajdującą się na pierwszym dysku. Niestety w ten sposób nie jesteśmy w stanie stwierdzić, czy jest to urządzenie SCSI, IDE czy SATA.

### 3.4.2 Zarządzanie partycjami

W dystrybucji RedHat istnieje kilka programów do zarządzania partycjami. Do najpopularniejszych należą: *fdisk*, *sfdisk* oraz *parted*. Na początku będziemy posługiwać się programem *fdisk*. Pierwsze jego uruchomienie ma zazwyczaj na celu sprawdzenie listy dostępnych urządzeń oraz ich podziału na partycje. Program *fdisk* uruchamiamy wówczas z opcją `-l`. W naszym przykładowym systemie wynik jej działania jest następujący:

```

1 [root@messy ~]# fdisk -l
2
3 Disk /dev/sda: 10.2 GB, 10202050560 bytes
4 255 heads, 63 sectors/track, 1240 cylinders
5 Units = cylinders of 16065 * 512 = 8225280 bytes
6 Disk identifier: 0xcd9acd9a
7
8      Device Boot      Start         End      Blocks   Id  System
9  /dev/sda1  *           1           65       522081   83  Linux
10 /dev/sda2             66          587      4192965   83  Linux
11 /dev/sda3           588          848      2096482+   83  Linux
12 /dev/sda4           849         1240      3148740    5  Extended
13 /dev/sda5           849         1109      2096451   83  Linux
14 /dev/sda6          1110         1174       522081   82  Linux swap / Solaris
15
16 Disk /dev/sdb: 10.2 GB, 10202050560 bytes
17 255 heads, 63 sectors/track, 1240 cylinders
18 Units = cylinders of 16065 * 512 = 8225280 bytes
19 Disk identifier: 0x90909090
20
21      Device Boot      Start         End      Blocks   Id  System
22  /dev/sdb1  *           1         1044      8385898+   83  Linux
23
24 Disk /dev/sdc: 15.3 GB, 15303075840 bytes
25 255 heads, 63 sectors/track, 1860 cylinders
26 Units = cylinders of 16065 * 512 = 8225280 bytes
27 Disk identifier: 0x000c83ce
28
29      Device Boot      Start         End      Blocks   Id  System
30

```

Informacja o dostępnych w systemie urządzeniach dyskowych dostarczana jest w formacie zwrotkowym. Każdą zwrotkę rozpoczyna informacja o nazwie urządzenia oraz jego pojemności. Kolejna linia to opis jego geometrii, a więc liczba głowic, sektorów oraz cylindrów. Następnie informacja o wielkości jednostki alokacji oraz identyfikator dysku. Treść tabelki zawierającej opis zajętości dysku pochodzi z tablicy alokacji dysku. W kolejnych kolumnach znajdują się:

- *Device* – nazwa pliku urządzenia.

- *Boot* – kolumna zawiera informację, czy na danej partycji znajdują się kod konieczny do inicjalizacji systemu operacyjnego. Partycje „bootowalne” zaznaczane są symbolem \*.
- *Start* – numer cylindra, od którego rozpoczyna się dana partycja.
- *End* – numer ostatniego cylindra budującego daną partycję.
- *Blocks* – rozmiar partycji w blokach. Po liczbie mogą pojawić się znaki + lub –. Oznaczają one zaokrąglenia rozmiaru partycji, odpowiednio w górę lub w dół, powstałe podczas tworzenia danej partycji gdy jej rozmiar podany np. w sektorach nie jest całkowitą wielokrotnością rozmiaru w cylindrach.
- *Id* – identyfikator określający jakiego typu systemu plików można spodziewać się na danej partycji.
- *System* – kolumna zawiera opis identyfikatora.

W naszym systemie znalezione zostały trzy dyski:

1. */dev/sda* – o pojemności 10.2 GB, posiadający 1240 cylindrów. Jeśli przyjrzymy się tabeli opisującej położenie partycji, to okaże się, że jego zasoby zostały prawie w całości wykorzystane. Utworzone zostały trzy partycje podstawowe: */dev/sda1* na cylindrach 1–65, */dev/sda2* na cylindrach 66–587 oraz */dev/sda3* na cylindrach 588–848. Wszystkie one posiadają identyfikator 83 co oznacza, że najprawdopodobniej znajduje się na nich natywny system plików Linuksa (któryś *ext*). Pozostała przestrzeń dysku zajmuje partycja rozszerzona */dev/sda4* utworzona na cylindrach 849–1240. Jej identyfikator to 5. Zostały na niej utworzone dwie partycje logiczne: */dev/sda5* na cylindrach 849–1109 oraz */dev/sda6* na cylindrach 1110–1174. Pierwsza z nich zawiera linuksowy system plików (identyfikator 83), zaś druga plik wymiany pamięci wirtualnej (identyfikator 82). Jak widać, istnieje jeszcze możliwość utworzenia partycji logicznych na cylindrach 1175–1240.
2. */dev/sdb* – o pojemności 10.2 GB, posiadający 1240 cylindrów. Została założona na nim jedna partycja */dev/sdb2* na cylindrach 1–1044. Jej identyfikator to 83 co oznacza, że znajduje się na niej linuksowy system plików. Pozostała część dysku, a więc cylindry 1045–1240 mogą zostać wykorzystane do utworzenia na nich kolejnych partycji.
3. */dev/sdc* – o pojemności 15.3 GB, posiadający 1860 cylindrów. Jego tablica alokacji jest pusta. Jest więc w całości dostępny.

Naukę korzystania z programu *fdisk* przeprowadzimy praktycznie. Założmy, że chcemy utworzyć partycję na urządzeniu */dev/sda* wykorzystując w tym celu pozostałe wolne miejsce. W tym celu uruchamiamy program *fdisk* podając jako argument nazwę pliku urządzenia:

```

1 [root@messy ~]# fdisk /dev/sda
2
3 The number of cylinders for this disk is set to 1240.
4 There is nothing wrong with that, but this is larger than 1024,
5 and could in certain setups cause problems with:
6 1) software that runs at boot time (e.g., old versions of LILO)
7 2) booting and partitioning software from other OSs
8    (e.g., DOS FDISK, OS/2 FDISK)
9
10 Command (m for help): m

```

```

11 Command action
12   a   toggle a bootable flag
13   b   edit bsd disklabel
14   c   toggle the dos compatibility flag
15   d   delete a partition
16   l   list known partition types
17   m   print this menu
18   n   add a new partition
19   o   create a new empty DOS partition table
20   p   print the partition table
21   q   quit without saving changes
22   s   create a new empty Sun disklabel
23   t   change a partition's system id
24   u   change display/entry units
25   v   verify the partition table
26   w   write table to disk and exit
27   x   extra functionality (experts only)
28
29 Command (m for help): x
30 Expert command (m for help): m
31 Command action
32   b   move beginning of data in a partition
33   c   change number of cylinders
34   d   print the raw data in the partition table
35   e   list extended partitions
36   f   fix partition order
37   g   create an IRIX (SGI) partition table
38   h   change number of heads
39   i   change the disk identifier
40   m   print this menu
41   p   print the partition table
42   q   quit without saving changes
43   r   return to main menu
44   s   change number of sectors/track
45   v   verify the partition table
46   w   write table to disk and exit
47
48 Expert command (m for help):
49

```

Po uruchomieniu program czyta tablicę partycji wskazanego urządzenia i przechowuje ją w swoich strukturach pamięci. Następnie wypisuje informację, że dysk posiada więcej niż 1024 cylindry, co nie jest niczym złym, ale może stanowić problem dla niektórych aplikacji czy programów ładujących system operacyjny. Jak widać w linii 10 program posiada swoją linię komend. Lista komend jest wypisywana komendą *m*. Przedstawia ją powyższy wydruk. Do najczęściej wykorzystywanych komend należą:

- *p* – komenda umożliwia wypisanie tablicy partycji znajdującej się aktualnie w strukturach pamięci programu.



- *n* – komenda umożliwia utworzenie nowej partycji.
- *d* – komenda umożliwia usunięcie istniejącej partycji.
- *t* – komenda służy do zmiany identyfikatora partycji.
- *l* – wypisuje listę możliwych identyfikatorów partycji.
- *q* – powoduje zakończenie pracy programu, ale bez zapisania zmian wprowadzonych we wczytanie tablicy partycji.
- *w* – kończy pracę programu i zapisuje tablicę partycji na dysk.
- *x* – umożliwia dostęp do rozszerzonej listy opcji programu.

Opcje rozszerzone programu *fdisk* wykorzystywane są rzadko, głównie po awarii dysku przy próbach odzyskania zapisanych na nim danych. Dlatego omówienie ich wraz z praktycznymi przykładami zostawimy na później.

Przystąpimy do utworzenia partycji na dysku */dev/sda*. W tym celu uruchamiamy program *fdisk* z argumentem będącym nazwą pliku urządzenia, na którym partycja ma zostać utworzona. Mamy zatem:

```

1 [root@messy ~]# fdisk /dev/sda
2
3 The number of cylinders for this disk is set to 1240.
4 There is nothing wrong with that, but this is larger than 1024,
5 and could in certain setups cause problems with:
6 1) software that runs at boot time (e.g., old versions of LILO)
7 2) booting and partitioning software from other OSs
8    (e.g., DOS FDISK, OS/2 FDISK)
9
10 Command (m for help): n
11 First cylinder (1175-1240, default 1175):
12 Using default value 1175
13 Last cylinder or +size or +sizeM or +sizeK (1175-1240, default 1240):
14 Using default value 1240

```

W linii komend programu posłużono się komendą *n* (linia 10). Program zażądał podania numeru pierwszego cylindra partycji. Zakres został podany w nawiasie. Wartością domyślną, wprowadzaną po naciśnięciu klawisza Enter, jest numer pierwszego wolnego cylindra. Jak widać w linii 11 to ona została wybrana. Następnie musimy określić rozmiar tworzonej partycji. Możemy tego dokonać na kilka sposobów. Najprostszy polega na podaniu numeru cylindra, który jako ostatni będzie wykorzystany do utworzenia partycji. Wówczas podajemy liczbę z zakresu podanego w nawiasie (linia 13). Wartością domyślną jest numer ostatniego dostępnego cylindra, którą można wprowadzić naciskając jedynie klawisz Enter. Miało to miejsce w naszym przypadku. Innym sposób zdefiniowania rozmiaru partycji polega na podaniu liczby cylindrów, które mają zostać użyte do jej utworzenia. Liczba musi zostać poprzedzona znakiem *+*. Znając numer pierwszego cylindra oraz ich liczbę program wyliczy numer ostatniego. Jednak podawania rozmiaru w cylindrach jest mało intuicyjne. Stąd kolejna możliwość polega na podaniu rozmiaru partycji w kB lub MB. Wówczas liczba będąca rozmiarem musi zostać poprzedzona znakiem *+*, a za nią musi pojawić się oznaczenie jednostki, odpowiednio *K* lub *M*. Przykładowo napis *+500M*

oznacza polecenie utworzenia partycji o rozmiarze 500MB. Program przeliczy podany rozmiar na liczbę cylindrów, dopełniając ich liczbę do całkowitej wielokrotności i obliczy numer ostatniego cylindra.

W chwili obecnej tablica partycji wygląda tak, jak przedstawia to poniższy listing:

```

1 Command (m for help): p
2
3 Disk /dev/sda: 10.2 GB, 10202050560 bytes
4 255 heads, 63 sectors/track, 1240 cylinders
5 Units = cylinders of 16065 * 512 = 8225280 bytes
6 Disk identifier: 0xcd9acd9a
7
8      Device Boot      Start         End      Blocks   Id  System
9  /dev/sda1    *           1           65       522081   83  Linux
10 /dev/sda2             66          587      4192965   83  Linux
11 /dev/sda3          588          848      2096482+   83  Linux
12 /dev/sda4          849         1240      3148740    5  Extended
13 /dev/sda5          849         1109      2096451   83  Linux
14 /dev/sda6        1110         1174       522081   82  Linux swap / Solaris
15 /dev/sda7        1175         1240      530113+   83  Linux

```

Jak widać utworzona została partycja, reprezentowana przez plik o nazwie `/dev/sda7`. Jej rozmiar to 530113 bloków, a zajmuje ona cylindry o numerach od 1175 do 1240. Partycji nadany został domyślny identyfikator 83. Jeśli, z różnych powodów, identyfikator ten nie jest odpowiedni to może zostać zmieniony. Przedstawiono to poniżej:

```

1 Command (m for help): t
2 Partition number (1-7): 7
3 Hex code (type L to list codes): l
4
5  0  Empty                1e  Hidden W95 FAT1 80  Old Minix        be  Solaris boot
6  1  FAT12                24  NEC DOS          81  Minix / old Lin bf  Solaris
7  2  XENIX root           39  Plan 9           82  Linux swap / So c1  DRDOS/sec (FAT-
8  3  XENIX usr            3c  PartitionMagic  83  Linux            c4  DRDOS/sec (FAT-
9  4  FAT16 <32M           40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
10 5  Extended             41  PPC PReP Boot   85  Linux extended  c7  Syrinx
11 6  FAT16                42  SFS             86  NTFS volume set da  Non-FS data
12 7  HPFS/NTFS            4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
13 8  AIX                 4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
14 9  AIX bootable         4f  QNX4.x 3rd part 8e  Linux LVM        df  BootIt
15 a  OS/2 Boot Manag     50  OnTrack DM      93  Amoeba          e1  DOS access
16 b  W95 FAT32           51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
17 c  W95 FAT32 (LBA)     52  CP/M            9f  BSD/OS          e4  SpeedStor
18 e  W95 FAT16 (LBA)     53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
19 f  W95 Ext'd (LBA)     54  OnTrackDM6      a5  FreeBSD         ee  EFI GPT
20 10 OPUS                55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
21 11 Hidden FAT12        56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
22 12 Compaq diagnost    5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
23 14 Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f4  SpeedStor

```

```

24 16 Hidden FAT16      63  GNU HURD or Sys ab  Darwin boot      f2  DOS secondary
25 17 Hidden HPFS/NTF 64  Novell Netware b7  BSDI fs          fd  Linux raid auto
26 18 AST SmartSleep 65  Novell Netware b8  BSDI swap        fe  LANstep
27 1b Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid ff  BBT
28 1c Hidden W95 FAT3 75  PC/IX
29 Hex code (type L to list codes): 8e
30 Changed system type of partition 7 to 8e (Linux LVM)

```

Zmiany identyfikatora partycji możemy dokonać uruchamiając w linii komend programu *fdisk* komendę *t* (linia 1). Następnie pojawia się pytanie o numer partycji (linia 2) oraz o nowy identyfikator (linia 3). Komenda *l* pozwala na wypisanie listy wartości oraz opisu dostępnych identyfikatorów. Pojawia się ona w postaci 4 kolumn. Pole lewe każdej z nich oznacza numer podany szesnastkowo, zaś prawe to opis. Podajemy żądany identyfikator (linia 29), zaś program informuje nas, że identyfikator wskazanej partycji został ustawiony (linia 30).

Przypomnijmy, że wszystkie wprowadzone w tablicy partycji zmiany dotyczą jedynie jej kopii przechowywanej w strukturach danych programu *fdisk*. Mamy teraz dwie możliwości: zakończyć pracę programu *fdisk* bez zapisu zmodyfikowanej tablicy partycji lub z zapisem. Pierwsze rozwiązanie wybieramy w przypadku, jeśli „pogubiliśmy” się z zmianach i lepiej będzie zacząć zmiany jeszcze raz od stanu początkowego. Jeśli pewni jesteśmy wprowadzonych zmian, to wybieramy rozwiązanie drugie, co przedstawiono poniżej:

```

1 Command (m for help): w
2 The partition table has been altered!
3
4 Calling ioctl() to re-read partition table.
5
6 WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
7 The kernel still uses the old table.
8 The new table will be used at the next reboot.
9 Syncing disks.

```

Zwróćmy uwagę, że jądro systemu operacyjnego będzie w dalszym ciągu używać poprzedniej tablicy partycji. To sytuacja normalna w przypadku dysków na których znajduje się partycja z kodem inicjalizującym systemu lub plikiem wymiany. Jeśli dla dalszych prac konieczne jest, aby jądro pracowało na aktualnej tablicy partycji, to możemy skorzystać z jednej z dwóch możliwości. Pierwsza, polegająca na ponownym uruchomieniu systemu, zadziała w każdym systemie operacyjnym. Druga dotyczy rodziny RedHat, w której mamy do dyspozycji komendę *partprobe*. Informuje ona jądro systemu o zmianach w tablicy partycji. W najprostszym przypadku może ona zostać uruchomiona bez opcji i argumentów. W takim przypadku zostaną przeglądnięte tablice partycji wszystkich dysków w systemie. Jeśli zmiany zostały wprowadzone w tablicy partycji jednego dysku, to nazwa reprezentującego go pliku może pojawić się jako argument wywołania komendy. Komenda *partprobe* posiada dwie istotne opcje: **-d** powoduje, że tablica partycji przechowywana w jądrze systemu nie zostanie zmodyfikowana, **-s** umożliwia wypisanie listy dostępnych urządzeń dyskowych wraz ze znajdującymi się na nich partycjami:

```

1 [root@messy ~]# partprobe -s
2 /dev/sda: msdos partitions 1 2 3 4 <5 6 7>
3 /dev/sdb: msdos partitions 1
4 /dev/sdc: msdos partitions

```

W przedstawionym przykładzie program *fdisk* został użyty do utworzenia partycji na dysku, na którym istniały już trzy partycje podstawowe (numer 1, 2 oraz 3 w linii 2 powyższego listingu) oraz partycja rozszerzona (numer 4), a na niej trzy partycje logiczne (o numerach 5, 6 i 7). Stąd nie pojawiło się pytanie o rodzaj partycji. Zwróćmy uwagę, że na urządzeniu */dev/sdc* nie została utworzona żadna partycja. Utwórzmy zatem dwie partycje: podstawową o rozmiarze 4 GB oraz logiczną na pozostałym obszarze dysku:

```

1  [root@messy ~]# fdisk /dev/sdc
2
3  Command (m for help): n
4  Command action
5      e   extended
6      p   primary partition (1-4)
7  p
8  Partition number (1-4): 1
9  First cylinder (1-1860, default 1):
10 Using default value 1
11 Last cylinder or +size or +sizeM or +sizeK (1-1860, default 1860): +4096M
12
13 Command (m for help): n
14 Command action
15      e   extended
16      p   primary partition (1-4)
17 e
18 Partition number (1-4): 2
19 First cylinder (500-1860, default 500):
20 Using default value 500
21 Last cylinder or +size or +sizeM or +sizeK (500-1860, default 1860):
22 Using default value 1860
23
24 Command (m for help): p
25
26 Disk /dev/sdc: 15.3 GB, 15303075840 bytes
27 255 heads, 63 sectors/track, 1860 cylinders
28 Units = cylinders of 16065 * 512 = 8225280 bytes
29 Disk identifier: 0x000c83ce
30
31      Device Boot      Start         End      Blocks   Id  System
32  /dev/sdc1              1           499       4008186    83  Linux
33  /dev/sdc2             500        1860      10932232+    5  Extended

```

Po uruchomieniu programu *fdisk* wybrano komendę *n* (linia 3). Ponieważ mamy dowolność jak chodzi o rodzaj tworzonej partycji program pyta, czy ma być to partycja podstawowa czy rozszerzona (linie 4–6). Wybrano typ podstawowy (linia 7, litera *p*). Nadajemy jej numer 1 (linia 8). Następnie odpowiadamy na znane już pytania dotyczące położenia partycji na dysku i jej rozmiaru. Zaczynamy od cylindra pierwszego (linie 9–10), zaś numer ostatniego cylindra zostanie wyliczony przez program, gdyż podajemy żądany rozmiar partycji pisząc *+4096M* (linia 11). Tworzymy drugą partycję, która będzie partycją rozszerzoną (linia 15–19, litera *e*). Nadajemy jej numer 2 (linia 20). Kwestię położenia na dysku oraz rozmiar rozwiązujemy wybierając ustawienia

domyślne (linie 21–24). Sprawdzamy tablicę partycji dysku (linia 24). Jak widać znajdują się na nim dwie partycje: numer 1, typu podstawowego na cylindrach 1–499 oraz numer 2, rozszerzoną, na cylindrach 500–1860).

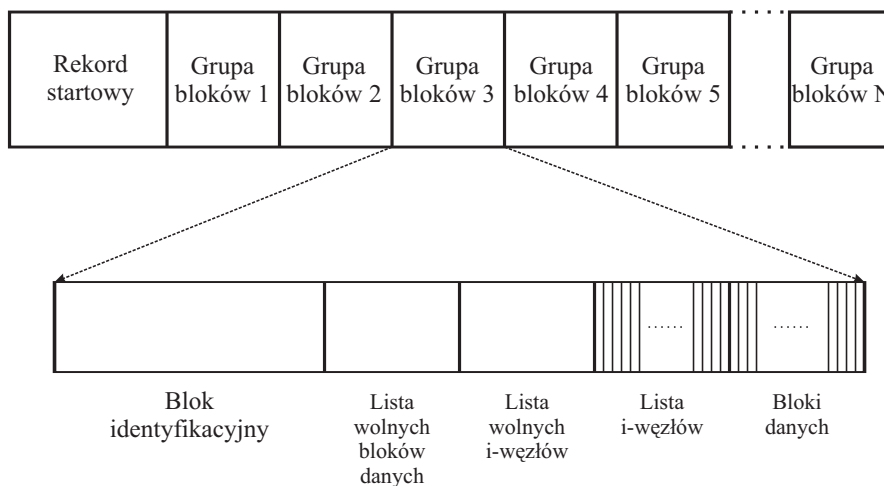
Reasumując, jeśli liczba partycji na dysku ma być większa niż cztery, to musimy stworzyć partycję rozszerzoną, aby na niej wydzielić partycje logiczne. Liczba partycji podstawowych może wówczas wynosić maksymalnie trzy. Przy liczbie partycji nie większej od czterech mamy swobodę w wyborze ich rodzaju.

### 3.4.3 System plików ext3

System plików ext3 (*Third Extended Filesystem*) stanowi rozwinięcie stabilnego i dobrze udokumentowanego systemu plików ext2. Rozwinięcie to polegało przede wszystkim na dodaniu funkcji kroniki co wymusiło zmiany w implementacji funkcji zapisu informacji do systemu plików. Jednak i–węzły oraz inne struktury danych są w systemie plików ext3 takie same jak w ext2. Umożliwia to prostą konwersję istniejącego systemu plików ext2 do ext3, jak również stosowanie tych samych programów narzędziowych.

#### Budowa wewnętrzna

Inspiracją dla twórców systemu ext2 stanowił system plików FFS, który dzięki koncepcji grup cylindrów zapewnia bliskość i–węzłów i odpowiadających im bloków danych, co ogranicza ruchy głowicą i powoduje zwiększenie wydajności operacji wejścia/wyjścia. Każda grupa bloków stanowi jakby autonomiczny system plików. Zawiera on oczywiście listę i–węzłów i bloki danych oraz struktury przyspieszające ich znajdowanie w postaci list wolnych i–węzłów oraz bloków danych. Każda grupa zawiera także kopię istotnych dla działania całego systemu plików informacji w postaci bloku opisu systemu plików. Rozwiązanie to w istotny sposób podnosi bezpieczeństwo systemu plików, gdyż uszkodzenie jednego bloku nie powoduje utraty danych przechowywanych w systemie plików (korzysta się wówczas z informacji zapisanej w innych superblokach).



Rysunek 3.12: System plików ext3 realizuje koncepcję grup bloków, analogiczną do grup cylindrów z systemu FFS. Bliskie sąsiedztwo i–węzłów oraz bloków danych ma na celu minimalizację ruchów głowicy, co prowadzi do zwiększenia wydajności operacji zapisu i odczytu danych z systemu plików.

Do rozwiązań zastosowanych w systemie plików ext2, a zwiększających jego wydajność należy:

- Zorganizowanie katalogów w postaci list odsyłaczowych wpisów. Każdy wpis zawiera numer i-węzła, informację o długości wpisu oraz nazwę opisywanego pliku lub katalogu. Takie rozwiązanie pozwala stosować nazwy o zróżnicowanych długościach bez marnowania miejsca w systemie plików. Domyślne ustawienia dopuszczają maksymalną długość nazw wynoszącą 255 znaków. W razie potrzeby ograniczenie to może zostać zwiększone do 1012 znaków.
- Algorytm obsługi pamięci podręcznej realizujący odczyt z wyprzedzeniem (*readaheads*). Polega on na tym, że po przeczytaniu żadanego bloku danych jądro wczytuje do pamięci podręcznej kilka kolejnych. Rozwiązanie to sprawdza się w przypadku sekwencyjnego odczytu pliku lub czytania zawartości katalogu.
- Optymalizacja przydzielania bloków danych. Na najwyższym poziomie pojawiły się grupy bloków zapewniające bliskie sąsiedztwo i-węzłów i wskazywanych przez nie bloki danych. Głównym kryterium przydzielania bloków danych jest aby znajdowały się one w tej samej co i-węzeł grupie bloków. Dodatkowo zapisywanie informacji do pliku powoduje, że system plików ext3 przydziela do 8-miu sąsiednich bloków danych. Niewykorzystane wracają do listy wolnych. Taki sposób wykorzystania bloków danych podnosi efektywność pamięci podręcznej.

Mechanizm dziennikowania został zaimplementowany w systemie plików ext3 jako oddzielny podsystem jądra, nazwany urządzeniem blokowym z kroniką (*journaling block device, JBD*). Podsystem ten udostępnia funkcje rejestrowania transakcyjnego podobne do wykorzystywanych w bazach danych. Z powodów wydajnościowych wykorzystuje on mechanizm opóźnionego zapisu, polegający na buforowaniu transakcji związanych z każdą aktualizacją stanu systemu plików. Zapewnia jednak, że w dzienniku zostaną zapisane wszystkie transakcje znajdujące się w buforze albo żadna z nich. Możliwe jest zrezygnowanie z mechanizmu opóźnionego zapisu w przypadku katalogów, które zostały oznaczone jako wymagające przeprowadzania operacji w trybie synchronicznym. Oznaczanie katalogów dokonuje się „ręcznie” z wykorzystaniem komendy *chattr* użytej z opcją **-S** i argumentem będącym nazwą katalogu.

Podsystem JBD udostępnia generalnie trzy tryby rejestrowania:

1. Rejestrowanie wszystkich zmian danych i metadanych (tryb *journaling*). Jest bezpieczne, gdyż w dzienniku przechowywany jest pełny zapis zmian informacji przeprowadzony w systemie plików. Za bezpieczeństwo płacimy jednak obniżeniem wydajności wynikającym z konieczności zapisywania zmian w dzienniku i zmian w systemie plików.
2. Rejestrowanie zmian wyłącznie metadanych (tryb *writeback*).
3. Rejestrowanie pełnych zmian bez spadku wydajności (tryb *ordered*). Tryb ten polega na zapisie na dysk danych związanych z transakcją przed aktualizacją metadanych. Dzięki zagwarantowaniu uaktualnienia danych przed zapisem na dysk informacji o zmianach w metadanych, związanych z zatwierdzoną transakcją, mamy pewność, że dane znajdujące się w plikach są po ich odtworzeniu z wykorzystaniem dziennika zawsze spójne z metadanymi systemu plików.

System plików ext3 wprowadza również dwa nowe typy plików, które po raz pierwszy pojawiły się w wersji 4.4 systemu operacyjnego BSD. Pliki stałe (*immutable*) są dostępne jedynie do odczytu. Nikt nie może modyfikować ich zawartości lub ich usunąć. Pliki tylko do dopisywania (*append-only*) mogą zostać otwarte w trybie do zapisu, ale zapisywane dane zostaną dopisane

na koniec pliku. Podobnie jak pliki stałe nie mogą zostać usunięte lub przeniesione. Są one wykorzystywane najczęściej jako pliki dziennika, których rozmiar może jedynie rosnąć.

Krokiem w stronę podniesienia bezpieczeństwa jest zarezerwowanie pewnej liczby bloków danych do dyspozycji jedynie użytkownika *root*. Domyślnie jest to 5% ich liczby w systemie plików. Umożliwią one prace administracyjne, np. odzyskiwanie danych, nawet w przypadku jeśli użytkownicy całkowicie go zapelnili. Dla systemów plików o dużych rozmiarach zaleca się ograniczenie tej wartości do 0.5–2%.

Ciekawym rozwiązaniem jest atrybut pozwalający w bezpieczny sposób usuwać pliki. Rozwiązanie polega na zapisaniu do bloków danych należących do usuwanego pliku bajtów o wartościach losowych. Dzięki temu jego zawartość nie będzie mogła zostać odczytana nawet programem służącym do edycji zawartości dysku.

W systemie ext3 zostały zaimplementowane tzw. *szybkie* dowiązania symboliczne. Dowiązania takie nie wykorzystuje bloków danych, gdyż nazwa pliku na które dowiązanie pokazuje nie jest przechowywana w bloku danych, a w i-węźle. Rozwiązanie to niewątpliwie prowadzi do oszczędności w blokach danych, jednak zwiększa rozmiar i-węzła. Stąd pojawiło się ograniczenie na długość nazwy do 60-ciu znaków.

Zwiększenie wydajności systemu plików możemy również osiągnąć zwiększając rozmiar bloku danych. Dzięki temu maleje liczba bloków danych koniecznych do przechowania informacji zapisanej w pliku. To z kolei zmniejsza liczbę operacji odczytu lub zapisu powodując, że liczba ruchów głowicy przy dostępie do danych staje się mniejsza, powodując wzrost wydajności. Stosowanie bloków danych o dużych rozmiarach powoduje jednak marnotrawienie miejsca w blokach danych. Rzadko zdarza się, aby rozmiar pliku stanowił całkowitą wielokrotność rozmiaru bloku danych. Dlatego blok przechowujący ostatni fragment danych nie jest w pełni wykorzystany. Im większy rozmiar bloku, tym ilość zmarnowanego miejsca jest większa. Typowe rozmiary bloków danych to 1024, 2048 oraz 4096 bajtów.

### Zakładanie systemu plików ext3

Generalnie do zakładania systemu plików w systemach linuxowych służy komenda *mkfs*. Opcja **-t** pozwala na zdefiniowanie typu zakładanego systemu. W najprostszym przypadku komenda wymaga podania jednego argumentu, którym jest ścieżka dostępu do pliku urządzenia, na którym system ten ma zostać założony. W tym przypadku wywołanie komendy *mkfs* będzie następujące:

```
1 [root@messy ~]# mkfs -t ext3 /dev/sda7
```

W rzeczywistości uruchamiany jest program znajdujący się w katalogu */sbin* o nazwie *mkfs.<typ>*, gdzie napis *<typ>* oznacza wyspecyfikowany po opcji **-t** typ systemu plików. W tym przypadku będzie to program *mkfs.ext3*. Przebieg jego wykonania jest następujący:

```
1 mke2fs 1.40.8 (13-Mar-2008)
2 Warning: 256-byte inodes not usable on older systems
3 Filesystem label=
4 OS type: Linux
5 Block size=4096 (log=2)
6 Fragment size=4096 (log=2)
7 33200 inodes, 132528 blocks
8 6626 blocks (5.00%) reserved for the super user
9 First data block=0
10 Maximum filesystem blocks=138412032
```

```
11 5 block groups
12 32768 blocks per group, 32768 fragments per group
13 6640 inodes per group
14 Superblock backups stored on blocks:
15     32768, 98304
16
17 Writing inode tables: done
18 Creating journal (4096 blocks): done
19 Writing superblocks and filesystem accounting information: done
20
21 This filesystem will be automatically checked every 27 mounts or
22 180 days, whichever comes first. Use tune2fs -c or -i to override.
```

Najistotniejsze parametry utworzonego systemu to:

- Rozmiar bloku danych wynosi 4096B (linia 5).
- System plików posiada 33200 i-węzłów oraz 132528 bloków danych (linia 7).
- 6626 bloków danych (5%) nie jest dostępnych dla procesów zwykłych użytkowników, a jedynie dla procesów użytkownika *root* (linia 8).
- System plików zawiera 5 grup bloków. Każda grupa zawiera 32768 bloków oraz 6640 i-węzłów (linie 11–13).
- Kopie bloku identyfikacyjnego systemu plików zostały zapisane w blokach numer 32768 oraz 98304 (linie 14–15).
- Utworzony został dziennik systemu plików o rozmiarze 4096 bloków danych. Program *mkfs.ext3* wykorzystuje następujące, domyślne rozmiary dzienników:
  - 1024 bloki jeśli system plików posiada mniej niż 32768 bloków.
  - 4096 bloków jeśli system plików zawiera co najmniej 32768 bloków ale nie mniej niż 262144 bloki.
  - 8192 bloki jeśli system plików zawiera co najmniej 262144 bloki.
- Sprawdzanie spójności systemu plików będzie odbywało się co 27 montowań lub co 180 dni, w zależności od tego co zajdzie wcześniej. Wartości te można zmodyfikować przy pomocy programu *tune2fs*.

Wartości atrybutów systemu plików zostały wyznaczone na podstawie rozmiaru partycji, na której jest on zakładany oraz średnich wartości zapewniających jego działanie w przeciętnych zastosowaniach. Często zdarza się jednak, iż tworzony system plików powinien, ze względu na przeznaczenie, posiadać szczególne właściwości. Ponieważ większość z nich nie może być zmieniona w istniejącym systemie plików, stosowne wartości należy nadać w trakcie jego tworzenia. Przykładowo, jeśli zależy nam, aby efektywność operacji wejścia/wyjścia była większa to, jak wiemy, najprostszym sposobem jest zwiększenie rozmiaru bloku danych. Jeśli system jest przeznaczony do przechowywania niewielkiej liczby plików o dużych rozmiarach, to wówczas można określić mniejszą od domyślnej liczbę dostępnych w nim i-węzłów. Pozwoli to dodatkowo na zwiększenie liczby bloków danych. W przeciwnym przypadku tzn. gdy w systemie plików przechowywane będą duże ilości małych plików, zwiększamy długość liczby i-węzłów kosztem liczby



bloków danych. Dodatkowo, nie zmieniamy domyślnego rozmiaru bloku danych, aby ilość traczonego miejsca w blokach przechowujących ostatnie fragmenty plików była minimalna. Rozmiar dostępnego dla użytkowników miejsca w systemach plików z rodziny ext można powiększyć, zmniejszając miejsce zarezerwowane jedynie dla użytkownika root. Domyślnie wynosi ono 5%, co przy rozmiarach współczesnych dysków daje duże wartości. Rozmiar przestrzeni zarezerwowanej można ograniczyć do, jak wspomniano, 1–2%. Zmiany wartości atrybutów dokonujemy w trakcie zakładania systemu plików, wykorzystując w tym celu dostępne opcje. Podstawowe opisano w podrozdziale dotyczącym systemu plików ext4, gdyż w chwili powstawania niniejszego opracowania, ten system plików był „najmłodszym” w rodzinie ext. W kolejnym punkcie przedstawiono kwestie związane z konfiguracją dziennika w systemach plików rodziny ext.

### Określanie rozmiaru i położenia dziennika

Program *mkfs.ext3* posiada opcję **-J**, która umożliwia nadawie wartości parametrom dziennika. Do podstawowych należy jego rozmiar. Zasadniczo wykorzystuje się ustawienia domyślne. Przypomnijmy, że przestrzeń dziennika jest wykorzystywana do zapisywania transakcji systemu plików, a więc jest zwalniana i ponownie zapisywana po zakończeniu i utrwaleniu transakcji. Pełnienie dziennika skutkuje wstrzymaniem dostępu do systemu plików oraz podjęciem przez podsystem JBD próby utrwalenia i zwolnienia wszystkich zatwierdzonych do wykonania transakcji. Większy dziennik może być przydatny w przypadku rejestrowania zmian zarówno metadanych jak i danych lub wówczas, gdy w systemie realizowane są transakcje zapisujące lub zmieniające duże ilości danych.

Istotny wpływ na wydajność systemu plików ma położenie dziennika. Jeśli jest to tylko możliwe, system plików i jego dziennik należy umieszczać na różnych dyskach. Sposób ten wymaga uprzedniego formatowania urządzenia, na którym znajdował się będzie dziennik, a następnie podczas zakładania systemu plików komendą *mkfs.ext3* wskazanie jego położenia. W naszym przykładowym systemie na dysku *hdc* utworzymy programem *fdisk* partycję o rozmiarze 20 MB. Następnie komendą *mkfs* założymy na niej dziennik dla systemu plików *ext3*. Wymaga to użycia opcji **-t** po której podany zostanie typ systemu plików, opcji **-O** z wartością *journal\_dev* informującą, że będzie to dziennik oraz argumentem którym jest ścieżka dostępu do pliku urządzenia, na którym zakładamy dziennik:

```

1 [root@messy ~]# mkfs -t ext3 -O journal_dev /dev/sdc1
2 mke2fs 1.40.8 (13-Mar-2008)
3 Filesystem label=
4 OS type: Linux
5 Block size=1024 (log=0)
6 Fragment size=1024 (log=0)
7 0 inodes, 24064 blocks
8 0 blocks (0.00%) reserved for the super user
9 First data block=1
10 0 block group
11 8192 blocks per group, 8192 fragments per group
12 0 inodes per group
13 Superblock backups stored on blocks:
14
15 Zeroing journal device: done

```

Następnie na urządzeniu */dev/sda7* zakładamy system plików, wskazując położenie dziennika:

```

1 [root@messy ~]# mkfs -t ext3 -j -J device=/dev/sdc1 /dev/sda7
2 mke2fs 1.40.8 (13-Mar-2008)
3 Warning: 256-byte inodes not usable on older systems
4 Filesystem label=
5 OS type: Linux
6 Block size=1024 (log=0)
7 Fragment size=1024 (log=0)
8 33280 inodes, 530112 blocks
9 26505 blocks (5.00%) reserved for the super user
10 First data block=1
11 Maximum filesystem blocks=67895296
12 65 block groups
13 8192 blocks per group, 8192 fragments per group
14 512 inodes per group
15 Superblock backups stored on blocks:
16     8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409
17
18 Writing inode tables: done
19 Adding journal to device /dev/sdc1: done
20 Writing superblocks and filesystem accounting information: done
21
22 This filesystem will be automatically checked every 39 mounts or
23 180 days, whichever comes first. Use tune2fs -c or -i to override.

```

Istotna informacja pojawiła się w linii 19 powyższego listingu. Założony system plików będzie korzystał z dziennika znajdującego się na urządzeniu `/dev/sdc1`. Korzystając z programu *debugfs* możemy sprawdzić właściwości założonego systemu plików. W tym celu komendę należy uruchomić z opcją **-R** i wartością *features* podając jako argument nazwę pliku urządzenia:

```

1 [root@messy ~]# debugfs -R features /dev/sda7
2 debugfs 1.40.8 (13-Mar-2008)
3 Filesystem features: has_journal ext_attr resize_inode dir_index filetype sparse_super

```

### Montowanie systemu plików ext3

Z ogólnych opcji komendy *mount*, których należy unikać przy montowaniu systemu plików *ext3* należy wymienić opcję *sync*. Wymusza ona wykonywanie wszystkich operacji wejścia/wyjścia w sposób synchroniczny. Spowoduje to zgubienie głównej korzyści korzystania z systemu plików *ext3*, który w tym wypadku będzie mniej wydajnym od systemu plików *ext2* stosującego operacje opóźnionego zapisu zwiększającą wydajność. Jeżeli jednak synchroniczny zapis w przypadku pojedynczych plików lub katalogów jest konieczny, to lepiej skorzystać z komendy *chattr*.

Spośród dedykowanych systemowi plików *ext3* opcji montowania należy wymienić:

- **data**={*journal, ordered, writeback*} – opcja umożliwia określenie trybu rejestrowania zmian zachodzących w systemie plików. Dla przypomnienia tryb *journal* oznacza, że wszystkie zmiany danych są zatwierdzane w dzienniku systemu plików jeszcze przed ich utwaleniem w samym systemie plików. Tryb *ordered* gwarantuje utwalenie w systemie plików wszystkich aktualizacji danych, zanim związane z nimi aktualizacje metadanych zostaną zarejestrowane

w dzienniku. Z kolei w trybie *writeback* aktualizacje danych systemu plików mogą być opóźniane i przeprowadzane już po zatwierdzeniu w dzienniku odpowiadających im zmian metadanych.

- **journal=inum** – pozwala określić numer i-węzła systemu plików, który zostanie przypisany do dziennika w chwili jego tworzenia.
- **journal=update** – opcja dokonuje aktualizacji dziennika systemu plików do aktualnej wersji.
- **noload** – użycie opcji spowoduje, że podczas montowania nie zostaną uwzględnione informacje zapisane w dzienniku systemu plików. Mówiąc inaczej, zostanie on zamontowany tak, jakby jego dziennik był pusty.

### Podstawowe programy narzędziowe

Do podstawowych programów narzędziowych systemu plików ext3 należy zaliczyć:

- **chattr** – program ten służy do zmiany atrybutów plików i katalogów. Przy jego pomocy można zdefiniować synchroniczne operacje wejścia wyjścia dla wybranych plików i katalogów.
- **debugfs** – program jest właściwie edytorem systemu plików. Umożliwia analizę działania jego podstawowych funkcji oraz nadawanie wartości niektórym metadanom. Przykładowo można zmienić wartość zmiennej *needs\_recovery*, która jeśli jest ustawiony oznacza, że rozpatrywany system plików jest niespójny i należy sięgnąć do zawartości dziennika. Narzędzie to działa podobnie jak *fdisk*, a mianowicie posiada zbiór komend uruchamianych w jego linii poleceń. Komendy umożliwiają edycję i-węzłów plików regularnych, katalogów oraz plików urządzeń, tworzenie i usuwanie plików i katalogów, odzyskiwanie zawartości usuniętych plików regularnych, itd.
- **e2fsck** – narzędzie to służy weryfikacji spójności systemu plików. Podczas uruchamiania wymagane jest podanie ścieżki pliku do urządzenia na którym znajduje się system plików. Sprawdzenie spójności polega na zamontowaniu systemu plików w katalogu tymczasowym oraz wymuszeniu na podsystemie JBD odtworzenia dziennika. Następnie zerowany jest bit *needs\_recovery*. Jeśli próba montowania zakończy się powodzeniem, a system plików zostanie zaznaczony jako poprawny, program *fsck* kończy działanie. Jeśli natomiast system plików zostanie zamontowany, ale w dalszym ciągu jest oznaczony jako niepoprawny, program *fsck* rozpoczyna proces weryfikacji spójności plików w taki sposób, jak dla systemu plików bez kroniki. Przykładowo, dla systemu plików ext2, proces ten składa się z pięciu faz: testów:

1. W fazie pierwszej weryfikacji podlegają i-węzły. Sprawdzana jest poprawność atrybutów plików i katalogów stowarzyszonych z danym i-węzłem, poprawność bloków danych znajdujących się w tablicy alokacji pliku każdego i-węzła. Sprawdzana jest także ewentualność wskazywania danego bloku danych przez tablice alokacji znajdujące się w kilku i-węzłach. W trakcie realizacji tej fazy tworzone są struktury danych, tzw. mapy bitowe, informujące o: 1. zajętości i-węzła oraz o tym czy opisuje on plik czy katalog, 2. opisujące bloki danych systemu plików, w tym bloki wolne oraz wskazywane przez kilka i-węzłów, 3. zawierające listy bloków danych stowarzyszonych z każdym i-węzłem opisującym katalog.

2. Faza druga dotyczy wszystkich katalogów systemu plików, które zostały zapisane w mapie bitowej utworzonej w fazie pierwszej. Sprawdzane jest: poprawność nazwy katalogu oraz długość wpisu katalogowego, poprawność numerów i-węzłów znajdujących się we wpisie, czy numer wskazuje na rzeczywiście używany i-węzeł (z wykorzystaniem mapy bitowej zajętości i-węzłów utworzonej w fazie pierwszej) oraz poprawność wpisów w katalogach (w szczególności czy pierwszym wpisem jest `.`, a i-węzeł związany z tym wpisem jest i-węzłem samego katalogu oraz czy drugim wpisem w katalogu jest katalog nadrzędny symbolicznie opisany przez `..`).
3. Trzecia faza to weryfikacja struktury drzewa katalogów. W fazie drugiej stwierdzono spójność informacji na poziomie pojedynczego katalogu, z uwzględnieniem jedynie występowania wpisów definiujących katalog bieżący i nadrzędny. Faza bieżąca sprawdza poprawność tych wpisów, dokonując próby przejścia przez każdy katalog systemu plików. Technicznie polega to na oznaczeniu katalogu głównego systemu plików jako sprawdzonego, a następnie przejściu od każdego katalogu systemu plików do katalogu głównego przez inne katalogi wykorzystując sprawdzone w fazie drugiej wpisy. Jeśli wędrówka nie zakończy się w katalogu głównym lub w jej trakcie dowolny katalog zostanie odwiedzony więcej niż jeden raz, to program *fsck* odłącza i-węzeł tego katalogu od systemu plików i umieszcza go w przeznaczonym do tego celu katalogu *lost+found*.
4. W fazie czwartej sprawdzane są liczniki dowiązań wszystkich i-węzłów systemu plików. Wartości wyznaczone w fazie pierwszej są porównywane z wartościami uzyskanymi w fazach drugiej i trzeciej. Pliki, których wartość licznika dowiązań wynosi 0 są umieszczane w katalogu *lost+found*.
5. Faza piąta polega na porównaniu informacji o systemie plików zebranych podczas poprzednich faz z tymi zapisanymi w bloku identyfikacyjnym. W przypadku występowania różnic informacje zapisane na dysku są zastępowane tymi zebranymi w trakcie analizy.

Program *fsck* w zależności od użytych opcji może poprawiać automatycznie wykryte nie-spójności, pytać użytkownika o sposób poprawy lub jedynie informować o znalezionych nieprawidłowościach.

- *lsattr* – program służy do wypisywania atrybutów plików. Wykorzystywany przykładowo do stwierdzania, czy dany plik jest specjalnie traktowany przez system plików.
- *mke2fs* – umożliwia zakładanie systemu plików ext2 oraz ext3 (wymaga użycia opcji **-j**).
- *resize2fs* – program służy do zmiany rozmiaru systemu plików z rodziny ext (wymaga użycia opcji **-j**). Możliwe jest zarówno zwiększenie jak i zmniejszenie rozmiaru. Operację przeprowadza się na liście bloków danych, a rozmiar listy i-węzłów jest zmieniany zgodnie z wartością atrybutu określającego rozmiar pamięci przeznaczonej dla bloków danych przypadających na jeden i-węzeł. Przed zmianą rozmiaru systemu plików konieczne jest przeprowadzenie badania spójności systemu plików programem *fsck* uruchomionym w tym przypadku z opcją **-f**.
- *tune2fs* – program wykorzystywany jest do zmiany wartości atrybutów systemu plików z rodziny ext (wymaga użycia opcji **-j**). Przykładem może być zmiana częstotliwości uruchamiania programu *fsck* przy okazji montowania danego systemu plików czy zmiana identyfikatora systemu plików.
- *findsuper* – program znajduje rezerwowe bloki identyfikacyjne systemu plików. Informacja w nich zapisana jest wykorzystywana zazwyczaj wówczas, gdy główny blok identyfikacyjny

został uszkodzony i sprawdzenie spójności systemu plików oraz jego zamontowanie nie jest możliwe.

- *rescuept* – narzędzie to jest przeznaczone do odzyskiwania tablicy partycji uszkodzonego dysku. Uzyskane informacje mogą zostać wykorzystane przez programy *fdisk*, *sfdisk* do jej odbudowy.

Oprócz wymienionych programów narzędziowych, dostępnych jest wiele innych. Należą do nich programy

#### 3.4.4 System plików ext4

Cieszący się dużą popularnością system plików ext3, ze względu na pojawienie się wielu systemów plików wywodzących się z dystrybucji komercyjnych po prostu się zestarzał. Problemem stała się niższa w stosunku do konkurentów wydajność oraz mała pojemność systemu plików wynosząca maksymalnie 16 TB. Stosowanie go na nowoczesnych macierzach dyskowych, posiadających znacznie większe pojemności, stało się zatem nieuzasadnione. Pierwsze zmiany w systemie ext3 pojawiły się w roku 2006. Wprowadzały one numery bloków danych zapisywane z wykorzystaniem 48 bitów, co pozwoliło zakładać systemy plików na wolumenach o rozmiarze do 1024 PB. Jak się jednak okazało, przeprowadzenie weryfikacji systemu plików o takim rozmiarze, założonego na dyskach o maksymalnie dostępnej obecnie wydajności, programem *e2fsck* trwałoby ponad 100 lat. Stąd pojawiła się konieczność zmian struktur metadanych systemu plików. Pojawiła się m.in. możliwość tworzenia tzw. obszarów (*extent*), które tworzyły położone fizycznie obok siebie bloki danych. Przechowywanie w pamięci listy wolnych obszarów, tworzonej podczas montowania systemu plików, umożliwia szybką optymalizację rozmieszczenia danych pliku na dysku. Wpłynęło to korzystnie na zwiększenie wydajności systemu plików, zwłaszcza podczas odczytu dużych plików. Dalsze prace doprowadziły do stworzenia kolejnego systemu plików z rodziny ext. Po raz pierwszy jego obsługa została włączona w jądrze systemu operacyjnego w wersji 2.6.19. W wersji 2.6.27 pojawił się jako eksperymentalny. Jako stabilny jest dostępny w jądrze 2.6.28. Pierwszą wykorzystującą go dystrybucją było Ubuntu 9.04. W dystrybucji Fedora jest dostępny od wersji 11.

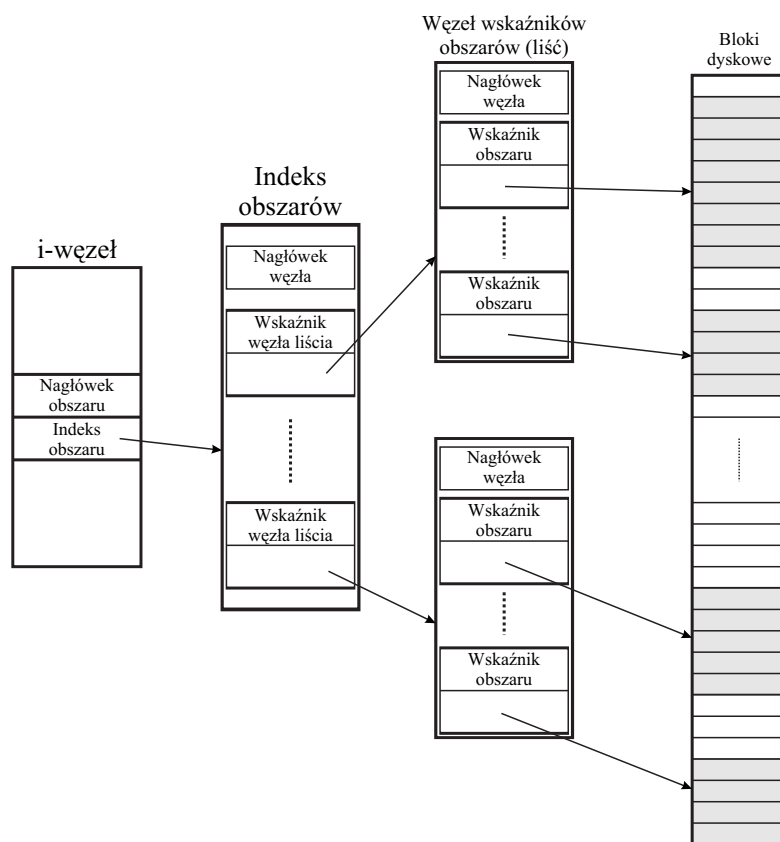
W kolejnym punkcie zostaną omówione podstawowe, zastosowane w nim rozwiązania, czyniące go konkurencyjnym dla innych rozwiązań systemem plików.

##### Budowa wewnętrzna

Zacznijmy od rozwiązania, które w innych systemach plików pojawiło się znacznie wcześniej. Mowa o obszarach bloków danych. W klasycznym rozwiązaniu (system plików s5), bloki danych przydzielane są pojedynczo. W tablicy alokacji pliku przechowywanej w i-węźle pamiętane są numery bloków danych przechowujących kolejne porcje danych pliku (rysunek 3.6). Modyfikacja pojawiła się w systemie plików XFS, stąd jej opis można znaleźć w podrozdziale poświęconym temu systemowi. W dużym skrócie polegała ona na wprowadzeniu algorytmu, który wyszukuje położone obok siebie bloki danych zwane obszarami. Zmianie uległy również wpisy w i-węźłach, które w tym rozwiązaniu zawierają numer pierwszego bloku danych oraz liczbę bloków tworzących dany obszar. Dodatkowo, podczas montowania systemu plików, generowana jest lista wolnych ekstendów dla każdej grupy bloków. Jest ona przechowywana w pamięci, a korzysta z niej podsystem alokacji bloków dla optymalizacji rozmieszczenia plików na dysku.

W systemie plików ext4 informacja o rozmieszczeniu pliku o rozmiarze do 512 MB jest przechowywana w i-węźle, w analogiczny sposób jak w systemie plików s5. Pozwala to na dostęp do danych zapisanych w pliku jedynie w oparciu o informacje w nim zapisane. Dla plików o rozmiarze

przekraczającym 512 MB tworzone jest drzewo opisujące obszary, przedstawione schematycznie na rysunku 3.13.



Rysunek 3.13: W wersji 4 systemu plików ext pojawiły się obszary. Są one adresowane z wykorzystaniem dwupoziomowego drzewa indeksów.

W zaproponowanej strukturze metadanych, pierwszy poziom stanowią indeksy obszarów. Struktura ta przechowuje numery bajtów w pliku, które rozpoczynają kolejne obszary oraz adresy tzw. węzłów wskaźników do obszarów. Węzły te zawierają numery bloków danych rozpoczynających obszar zawierający fragment informacji pliku oraz informację o liczbie bloków tworzących go. W przypadku dużych plików, wprowadzenie obszarów oraz opisujących je drzew rozwiązuje dwa problemy występujące w systemie ext3. Po pierwsze znacznie ogranicza zajętość miejsca wykorzystywanego przez metadane, po drugie zmniejsza defragmentację zwiększając wydajność systemu plików. Skraca również czas weryfikacji systemu plików programem *fsck*, gdyż umożliwia szybką detekcję oraz omijanie w tym procesie niewykorzystywanych obszarów dysku.

Poprawę efektywności uzyskano również stosując opóźnienie alokacji przestrzeni dyskowej dla danych zapisanych w plikach do momentu, w którym konieczne staje się odnotowanie związanej z nią transakcji przechowywanej w dzienniku. Dodatkowo, jeśli aplikacja wymaga zarezerwowania przestrzeni zanim zacznie jej używać, to system plików ext4 robi to, w przeciwieństwie do wielu innych systemów plików, bez fizycznego zapisywania bajtów zerowych do tego obszaru. Uzyskane w ten sposób zwiększenie wydajności jest widoczne zwłaszcza w aplikacjach bazodanowych oraz programach narzędziowych dla plików multimedialnych. Dostarczane przez Google łaty na jądro

od wersji 2.6.29 umożliwiają korzystanie z systemu ext4 pracującym bez dziennikowania, co podnosi jego wydajność kosztem bezpieczeństwa przechowywanych danych. Nie są jednak jeszcze dostępne programy narzędziowe dla pracującego w tym trybie systemu plików.

Wprowadzono także zmiany mające na celu podniesienia poziomu bezpieczeństwa systemu plików. Każda transakcja zostaje zapisana w dzienniku systemu plików wraz z sumą kontrolną. Pozwala to na szybką detekcję błędnych zapisów w dzienniku oraz przyspiesza odwzorowywanie transakcji zapisanych w dzienniku w systemie plików. Suma kontrolna jest także wykorzystywana w deskryptorach grup bloków.

System plików ext4 wykorzystuje także mechanizm bariery oferowany przez najnowsze urządzenia dyskowe. Jego działanie polega na przechowywaniu w pamięci podręcznej wielu operacji zapisu i ich posortowanie. Sterownik urządzenia realizuje operacje zapisu w kolejności wynikającej z przynależności do transakcji. Dzięki temu wszystkie zapisy związane z daną transakcją zostaną zapisane w systemie plików przed zaznaczeniem ich jako wykonane w dzienniku. Mechanizm ten jest domyślny. Jego wyłączenie można zrealizować wykorzystując podczas montowania opcję **barrier** z wartością 0.

Istotne jest także usunięcie kilku uciążliwych ograniczeń występujących w systemie ext3. Przykładowo liczba podkatalogów możliwych do utworzenia w katalogu w systemie ext3 wynosiła maksymalnie 32000. W systemie ext4 jest teoretycznie nieograniczona. Zmianie uległ również format zapisu wszystkich znaczników czasowych. Są one zapisywane z dokładnością do nanosekund (w systemie plików ext3 były to sekundy), zaś sposób reprezentacji umożliwia obsługę dat do 25 kwietnia 2514 roku (w systemie plików ext3 możliwości skończyły się dnia 18 stycznia 2038 roku).

### Zakładanie systemu plików ext4

Jednym ze sposobów założenia systemu plików ext4 jest dokonanie konwersji istniejącego systemu plików ext3. Dokonujemy jej korzystając z programu *tune2fs*, który dostępny jest w pakiecie *e2fsprogs*. Pakiet zawiera kod źródłowy programów narzędziowych dla systemów plików z rodziny ext. Zalecane jest korzystanie z ostatniej, stabilnej wersji, która jest dostępna np. na stronie <http://sourceforge.net/projects/e2fsprogs>. Kompilację oraz instalację programów narzędziowych i dokumentacji należy przeprowadzić zgodnie z instrukcjami zawartymi w pliku *INSTALL* znajdującym się w katalogu utworzonym po rozpakowaniu pakietu.

Dla dokonania konwersji program *tune2fs* uruchamiamy z argumentem będącym nazwą pliku reprezentującego urządzenie, na którym znajduje się konwertowany system plików. Wymagane jest użycie opcji **-O** z przełącznikami:

- *extents* – wprowadza zmianę w sposobie adresowania bloków danych. W dotychczasowym rozwiązaniu bloki były adresowane pojedynczo adresem bezpośrednim lub pośrednim (system plików *s5*). System plików ext4 wprowadza obszary bloków danych zawierające bloki o kolejnych numerach.
- *uninit\_bg* – umożliwia jądru systemu operacyjnego zainicjalizowania map bitowych i tablic i-węzłów oraz znaczników niewykorzystanych i-węzłów. Czynności te umożliwią szybsze działanie programu *e2fsck*.
- *dir\_index* – wymusza używanie B drzew do indeksowania plików w obrębie katalogów.

Tak użyty program wprowadza istotne zmiany w metadanych istniejącego systemu plików. Stąd konieczna jest weryfikacja jego spójności. Dokonujemy jej programem *e2fsck*, który uruchamiamy z argumentem będącym ścieżką dostępu do pliku reprezentującego urządzenie, na którym znajduje się system plików. Wymagane jest użycie następujących opcji:

- **-p** – wynikiem użycia opcji jest automatyczna korekta znalezionych nieprawidłowości.
- **-D** – opcja dokonuje reorganizacji katalogów pod kątem skrócenia czasu dostępu. Dodatkowo sprawdza, czy do danego katalogu nie prowadzi więcej niż jedna ścieżka dostępu.
- **-f** – wymusza weryfikację nawet, jeśli system plików jest zaznaczony jako czysty.

Niekiedy może się okazać, że znalezione niespójności będą wymagały uruchomienia programu *e2fsck* bez opcji. Przypadek ten przedstawiono poniżej:

```

1 [root@messy build]# tune2fs -O extents,uninit_bg,dir_index /dev/sda7
2 tune2fs 1.41.9 (22-Aug-2009)
3
4 Please run e2fsck on the filesystem.
5
6 [root@messy build]# e2fsck -pDf /dev/sda7
7 /dev/sda7: One or more block group descriptor checksums are invalid.  FIXED.
8 /dev/sda7: Group descriptor 0 checksum is invalid.
9
10 /dev/sda7: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.
11      (i.e., without -a or -p options)
12 [root@messy build]# e2fsck /dev/sda7
13 e2fsck 1.41.9 (22-Aug-2009)
14 One or more block group descriptor checksums are invalid.  Fix<y>? yes
15
16 Group descriptor 0 checksum is invalid.  FIXED.
17 Group descriptor 1 checksum is invalid.  FIXED.
18 Group descriptor 2 checksum is invalid.  FIXED.
19 Group descriptor 3 checksum is invalid.  FIXED.
20 Group descriptor 4 checksum is invalid.  FIXED.
21 /dev/sda7 contains a file system with errors, check forced.
22 Pass 1: Checking inodes, blocks, and sizes
23 Pass 2: Checking directory structure
24 Pass 3: Checking directory connectivity
25 Pass 4: Checking reference counts
26 Pass 5: Checking group summary information
27 /dev/sda7: 11/33200 files (0.0% non-contiguous), 6295/132528 blocks
28 [root@messy build]# e2fsck /dev/sda7
29 e2fsck 1.41.9 (22-Aug-2009)
30 /dev/sda7: clean, 11/33200 files, 6295/132528 blocks

```

Często istnieje jednak konieczność rozpoczęcia pracy z założonym jako ext4, a nie konwertowanym systemem plików. Programy narzędziowe, jeśli nie są dostępne w dystrybucji, można zainstalować z pakietu *e2fsprogs*. Domyślnie pliki zostaną skopiowane do katalogu */sbin*, a ich nazwy, zgodnie z konwencją, będą kończyły się na *.ext4*. Stąd założenie systemu plików ext4 najprościej można przeprowadzić uruchamiając program *mkfs* o opcją **-t** i wartością *ext4* oraz argumentem będącym nazwą pliku urządzenia, na którym system ten ma zostać założony. Przykładowo:

```

1 [root@messy sbin]# mkfs -t ext4 /dev/sda7
2 mke2fs 1.41.9 (22-Aug-2009)

```



### 3.4. ADMINISTRACJA WYBRANYMI SYSTEMAMI PLIKÓW W DYSTRYBUCJI REDHAT209

```
3 | Filesystem label=
4 | OS type: Linux
5 | Block size=4096 (log=2)
6 | Fragment size=4096 (log=2)
7 | 33200 inodes, 132528 blocks
8 | 6626 blocks (5.00%) reserved for the super user
9 | First data block=0
10 | Maximum filesystem blocks=138412032
11 | 5 block groups
12 | 32768 blocks per group, 32768 fragments per group
13 | 6640 inodes per group
14 | Superblock backups stored on blocks:
15 |     32768, 98304
16 |
17 | Writing inode tables: done
18 | Creating journal (4096 blocks): done
19 | Writing superblocks and filesystem accounting information: done
20 |
21 | This filesystem will be automatically checked every 34 mounts or
22 | 180 days, whichever comes first. Use tune2fs -c or -i to override.
```

Zwróćmy uwagę na wartości podstawowych atrybutów założonego systemu plików:

- Rozmiar bloku danych wynosi 4096 B (linia 5). Rozmiar fragmentu również został ustalony na 4096 B (linia 6).
- System plików został wyposażony w 33200 i-węzłów oraz 132528 bloki danych (linia 7). Wynika stąd, że na jeden i-węzeł będzie przypadało średnio 16384 B danych (czyli 4 bloki danych). Jeśli pliki w systemie będą posiadały rozmiar mniejszy, to jako pierwsze wyczerpią się i-węzły, a pozostaną wolne bloki danych. W przypadku plików większych sytuacja będzie odwrotna.
- 6626 bloków danych, co stanowi 5% wszystkich bloków danych zostało zarezerwowanych dla użytkownika *root* (linia 9). Będzie on miał możliwość prowadzenia prac administratorskich nawet w przypadku wykorzystania bloków danych przez użytkowników.
- Maksymalny rozmiar systemu plików może wynosić 138412032 bloków danych (linia 10), co przy rozmiarze bloku wynoszącym 4096 B daje ok. 528 GB. To istotna informacja dla przypadku, w którym system ten zostałby założony na wolumenie logicznym systemu LVM (podrozdział 3.6).
- System plików posiada 5 grup bloków (linia 11) (rysunek 3.12). W każdej grupie znajduje się 32768 bloków, 32768 fragmentów (linia 12), oraz 6640 i-węzłów (linia 13).
- Kopie bloków identyfikacyjnych znajdują się w blokach o numerach 32768 oraz 98304 (linie 14 i 15).
- Dziennik systemu plików został założony na tym samym urządzeniu, a jego rozmiar wynosi 4096 bloków danych (linia 18).

Program *mkfs.ext4* posiada wiele opcji. Krótki przegląd rozpoczniemy od umożliwiających ustalenie liczby i-węzłów i rozmiaru bloku danych. Należą do nich:

- **-i** *liczba\_bajtów\_na\_i-węzeł* – wartością opcji jest liczba bajtów bloków danych opisywanych przez jeden i-węzeł. Im wartość ta jest większa, tym mniejsza będzie liczba i-węzłów w zakładanym systemie plików. Zaleca się aby wartość ta nie była mniejsza od rozmiaru bloku danych. W przeciwnym przypadku liczba utworzonych i-węzłów będzie większa niż liczba bloków danych, co spowoduje, że część i-węzłów nie będzie nigdy wykorzystana.
- **-N** *liczba\_i-węzłów* – opcja umożliwia bezpośrednie podanie liczby i-węzłów, które mają zostać utworzone w zakładanym systemie plików. Wartość obliczona przez program na podstawie liczby bloków danych oraz wartości współczynnika określającego liczbę bajtów opisanych przez i-węzeł jest wówczas pomijana.
- **-b** *rozmiar\_bloku* – opcję wykorzystuje się do określenia rozmiaru bloku danych. Wartość podaje się w bajtach. Możliwe wartości to 1024, 2048 oraz 4096. Pominięcie opcji skutkuje określeniem rozmiaru bloku na podstawie rozmiaru systemu plików oraz sposobu jego wykorzystania (opcja **-T**). Jeśli wartość opcji została poprzedzona minusem (np. **-b -2048**), to wówczas program *mkfs.ext4* określa rozmiar bloku danych z dodatkowym warunkiem, że musi on wynosić co najmniej tyle ile moduł z podanej wartości. Jest to wykorzystywane w przypadku urządzeń wymagających, aby rozmiar bloku stanowił wielokrotność 2 kB.
- **-T** *opcja\_wykorzystania, ...* – opcję tą wykorzystujemy, jeśli potrafimy określić przeznaczenie systemu plików, ale nie chce się nam obliczać wartości jego atrybutów. Wartości opcji stanowią sugestie dla heurystyk wykorzystywanych przez program *mkfs.ext4* do określania wartości atrybutów. Przykładowo na partycji o rozmiarze 0.5 GB użycie wartości opcji *small* spowoduje utworzenie systemu plików z blokiem danych o rozmiarze 1024 B, 132600 i-węzłami oraz 530112 blokami danych. Wartość *floppy* umożliwi utworzenie systemu plików z identyczną liczbą bloków danych o takim samym rozmiarze, ale z 66560 i-węzłami.
- **-O** *cecha, ...* – opcja umożliwia utworzenie systemu plików wyposażonego w wyspecyfikowane cechy. W praktyce chodzi o nadanie odpowiednim atrybutom wartości, które zakładany system plików będą predystynowały do konkretnych zastosowań. Wartości domyślne atrybutów nadaje cecha *base\_features*. Są one zdefiniowane w pliku */etc/mke2fs.conf*, którego budowę omówiono poniżej. Cechę można poprzedzić znakiem  $\wedge$ , co oznacza jej negację. Użycie cechy *none* powoduje usunięcie wszystkich cech systemu plików. Możliwe do użycia cechy systemu ext4 to:
  - *large\_file* – system plików założony z tą cechą będzie mógł przechowywać pliki o rozmiarze większym niż 2 GB. Obecnie jądra systemu operacyjnego automatycznie ustawiają tę cechę w momencie przekroczenia przez plik rozmiaru 2 GB.
  - *dir\_index* – cecha pozwala na wykorzystywanie B drzew do indeksowania plików w katalogach w celu przyspieszenia ich wyszukiwania.
  - *filetype* – użycie cechy umożliwia przechowywanie informacji o typie pliku we wpisie definiującym go w katalogu. Przypomnijmy, że w systemie plików *s5* wpis zawierał nazwę i numer i-węzła pliku.
  - *flex\_bg* – cecha powoduje, iż mapy bitowe oraz tablice i-węzłów będą mogły zostać rozmieszczone w grupach położonych w dowolnym miejscu urządzenia fizycznego.
  - *has\_journal* – umożliwia utworzenie dziennika systemu plików (analogia do opcji **-j**). Cecha domyślna.

- *journal\_dev* – cecha umożliwia utworzenie dziennika systemu plików na oddzielnym urządzeniu. Jedynym ograniczeniem jest, że rozmiar bloku danych systemu plików i jego dziennika musi być taki sam.
  - *extent* – użycie cechy sprawia, że system plików stara się zajmować spójne obszary i-węzłów zamiast adresować je pojedynczo. Prowadzi to do zwiększenia wydajności systemu plików, zwłaszcza podczas odczytu dużych plików (odczyt całego obszaru bloków danych jest szybszy od odczytu pojedynczych, tworzących go bloków).
  - *uninit\_bg* – pozwala na założenie systemu plików, w którym nie wszystkie grupy bloków zostaną zainicjalizowane. Pozwala to na skrócenie czasu zakładania systemu plików, jak również badania jego spójności programem *e2fsck*.
  - *resize\_inode* – użycie cechy powoduje zarezerwowanie miejsca dla tablicy deskryptorów grup bloków danych na wypadek ewentualnego zwiększenia rozmiaru systemu plików. Domyślnie ilość zarezerwowanego miejsca umożliwia 1024-krotny wzrost w stosunku do rozmiaru pierwotnego.
  - *sparse\_super* – umożliwia utworzenie systemu plików ze zmniejszoną liczbą kopii bloków identyfikacyjnych w celu zaoszczędzenia miejsca w systemie plików.
- **-E** *opcja\_rozszerzona,...* – pozwala na nadanie wartości atrybutom określonym jako dodatkowe. Lista wartości składa się nazwy opcji oraz przypisanej jej wartości po znaku =. W przypadku kilku opcji separatorem jest „.”. Opcja ta stanowi rozwinięcie opcji **-R**, która w poprzednich wersjach programu służyła konfigurowaniu systemów plików ext2 oraz ext3 do pracy na macierzach dyskowych RAID (podrozdział 3.7). Stąd do najczęściej wykorzystywanych wartości opcji należą:
- *stride=rozmiar\_kawalka* – pozwala na skonfigurowanie systemu plików z przeznaczeniem dla macierzy RAID, pracującej z rozmiarem kawałka (*chunk*) zawierającym podaną liczbę bloków danych. Wartość określa maksymalną liczbę bloków danych po zapisaniu lub odczycie których z danego urządzenia fizycznego system zacznie wykorzystywać następne urządzenie fizyczne. Właściwość ta jest wykorzystywana do zapobiegania sytuacji, w której metadane systemu plików mogą zostać umieszczone na pojedynczym urządzeniu fizycznym, co może radykalnie zmniejszyć wydajność systemu plików. Wartość ta bywa też wykorzystywana do alokowania grupy bloków danych.
  - *stripe\_width=rozmiar\_paska* – umożliwia skonfigurowanie systemu plików do pracy z macierzą RAID posiadającą pasek zawierający podaną liczbę bloków danych systemu plików. Wartość ta typowo stanowi iloczyn rozmiaru kawałka oraz liczby urządzeń fizycznych *aktywnie* przechowujących dane. Przykładowo, w systemie RAID5 zbudowanym na  $N$  dyskach rozmiar ten wynosi  $(N - 1) \times \text{rozmiar\_kawalka}$ , gdyż rozproszona suma kontrolna zajmuje pojemność udostępnianą efektywnie przez jedno fizyczne urządzenie budujące macierz.
  - *resize=liczba\_bloków\_danych* – powoduje zarezerwowanie obszaru w systemie plików dla tablicy deskryptorów grup bloków danych tak, aby mogła ona obsługiwać system plików powiększony do zadanej liczby bloków danych.

Wartości domyślne atrybutów, które wykorzystuje program *mkfs.ext4* zakładanego systemu plików są zapisane w pliku konfiguracyjnym o nazwie */etc/mke2fs.conf*. Jego zawartość przedstawiono poniżej:

```

1  [defaults]
2      base_features = sparse_super,filetype,resize_inode,dir_index,ext_attr
3      blocksize = 4096
4      inode_size = 256
5      inode_ratio = 16384
6
7  [fs_types]
8      ext3 = {
9          features = has_journal
10     }
11     ext4 = {
12         features = has_journal,extents,huge_file,flex_bg,uninit_bg,dir_nlink,extra_is
13         inode_size = 256
14     }
15     ext4dev = {
16         features = has_journal,extents,huge_file,flex_bg,uninit_bg,dir_nlink,extra_is
17         inode_size = 256
18         options = test_fs=1
19     }
20     small = {
21         blocksize = 1024
22         inode_size = 128
23         inode_ratio = 4096
24     }
25     floppy = {
26         blocksize = 1024
27         inode_size = 128
28         inode_ratio = 8192
29     }
30     news = {
31         inode_ratio = 4096
32     }
33     largefile = {
34         inode_ratio = 1048576
35         blocksize = -1
36     }
37     largefile4 = {
38         inode_ratio = 4194304
39         blocksize = -1
40     }
41     hurd = {
42         blocksize = 4096
43         inode_size = 128
44     }
45

```

Jak widać plik ten ma budowę zwrotkową. Zwrotka z nagłówkiem `[defaults]` określa wartości atrybutów dla przypadku, w którym nie podano żadnych przesłanek co do cech systemu

plików i jego zastosowania oraz nie określono w inny sposób wartości atrybutów. Miało to miejsce w naszym przypadku, gdzie uruchamiając program *mkfs.ext4* określono jedynie nazwę pliku urządzenia na którym system plików ma zostać założony. Stąd wartości atrybutów utworzonego systemu plików były identyczne z zapisanymi w twj zwrotce. Kolejna zwrotka *fs\_types* określa cechy wybranego systemu plików. Jeśli zakładamy system plików ext3, to cecha *has\_journal* spowoduje utworzenie dziennika. W przypadku systemu plików ext4 lista cech jest znacznie dłuższa. Oprócz dziennika pojawi się administracja obszarami, obsługa dużych plików, itd. Rozmiar i-węzła będzie wynosił 256 B. Następnie pojawiły się wartości opcji **-T** definiujące rozmiar bloku danych, rozmiar i-węzła oraz liczbę bajtów opisywanych przez jeden i-węzeł.

Na zakończenie kilka opcji programu *mkfs.ext4*, które rzadziej aczkolwiek bywają użyteczne. Należą do nich:

- **-c** – użycie opcji wymusza tesowanie urządzenia, na którym ma zostać założony system plików pod kątem znalezienia bloków uszkodzonych. Jednokrotne użycie opcji powoduje wykonanie prostego testu odczytu, dwukrotne zaś testu odczytu–zapisu (test ten jest znacznie bardziej długotrwały).
- **-g *liczba\_bloków\_na\_grupę*** – użycie tej opcji umożliwia określenie liczby bloków budujących grupę. Jednak w przypadku systemu plików zakładanego na pojedynczych urządzeniach fizycznych w żadan sposób nie wpływa on na jego wydajność. W przypadku systemu plików zakładanego na macierzach RAID zaleca się użycie opcji **-E** umożliwiającej określenie rozmiaru kawałka. Opcja ta jest używana raczej w celach testowych.
- **-G *liczba\_grup*** – opcja umożliwia określenie liczby grup bloków, które zostaną umieszczone fizycznie obok siebie w celu utworzenia wirtualnego bloku grup systemu plików ext4. Liczba grup musi być wielokrotnością liczby 2 i może zostać efektywnie wprowadzona z użyciem opcji rozszerzonej *flex\_bg*.
- **-I *rozmiar\_i-węzła*** – opcja umożliwia ustalenie rozmiaru i-węzła wykorzystywanego w zakładanym systemie plików. Wartość domyślna to 256 B. Rozmiar musi stanowić potęgę liczby 2. Zwiększenie rozmiaru i-węzła pozwala przechowywać wartości dodatkowych atrybutów plików. Niestety nie będą one udostępniane przez jądra w wersji 2.4, jak również systemy plików wykorzystujące i-węzły o dużych rozmiarach nie będą mogły zostać przez nie zamontowane. Rozmiar i-węzła musi zostać ustalony podczas zakładania systemu plików, gdyż w już utworzonym systemie plików nie ma możliwości jego zmiany.
- **-L *etykieta\_systemu\_plików*** – wartością opcji jest etykieta, która zostanie nadana zakładanemu systemowi plików. Maksymalna długość wynosi 16 B.
- **-m *liczba\_bloków\_zarezerwowanych*** – umożliwia określenie liczby bloków danych zakładanego systemu plików zarezerwowanych do wykorzystania przez użytkownika *root*. Bywa wykorzystywana w dyżych systemach plików przeznaczonych do przechowywania niezbyt często zmieniających się danych użytkowników. Zauważmy, że wartość domyślna 5% przy rozmiarze systemu plików o rozmiarze 1 TB rezerwuje obszar 50 GB. W praktyce można go ograniczyć do 1-2% czyli 10-20GB.
- **-r *numer\_rewizji*** – wartością opcji jest tzw. numer rewizji (*revision number*). Starsze wersje jądra systemu operacyjnego obsługują jedynie numer 0. Wartością domyślną jest 1.
- **-t *typ\_systemu\_plików*** – dopuszczalne (w chwili powstawania niniejszego podręcznika) wartości opcji, to: *ext2*, *ext3*, *ext4*. Pozwalają one na wykorzystanie domyślnych wartości atrybutów dla zakładanego systemu plików, zdefiniowanych w opisanym powyżej pliku

*/etc/mke2fs.conf*. Należy zwrócić uwagę na możliwość takiego zdefiniowania wartości atrybutów tworzonego systemu plików, że nie będzie on obsługiwany przez kod zawarty w jądrze systemu operacyjnego. Pojawia się ona przy jednoczesnym użyciu opcji **-O**. Przykładem może być polecenie: *mke2fs -t ext3 -O extents /dev/sda7*, które utworzy system ext3, wyposażony w możliwość zajmowania spójnych obszarów bloków danych, co jak wiemy w tym systemie plików nie występuje. Z kolei polecenie: *mke2fs -t ext3 -O has\_journal /dev/sda7* doprowadzi do utworzenia systemu plików ext3 pozbawionego mechanizmu dziennikowania.

- **-T przeznaczenie\_systemu\_plików** – opcja pozwala na wybranie optymalnych dla danego przeznaczenia wartości atrybutów systemu plików. Zostały one zdefiniowane w pliku */etc/mke2fs.conf*. Pominięcie opcji powoduje, że wartości atrybutów zostają wyznaczone w oparciu o heurystyki, na podstawie rozmiaru zakładanego systemu plików. W przeciwnym przypadku, jeśli rozmiar ten jest mniejszy od 3MB, to polecenie *mke2fs* użyje wartości dla systemu typu *floppy*. Dla systemów o rozmiarach większych od 3MB, a mniejszych od 512MB użyte zostaną wartości dla typu *small*. W pozostałych przypadkach będą to wartości dla typu *default*.
- **-U UUID** – wartością opcji jest unikalny identyfikator globalny, który zostanie nadany zakładanemu systemowi plików.
- **-v** – przełącza tryb pracy polecenia na gadatliwy, ze szczegółową informacją o wykonywanych czynnościach.

#### Zmiana wartości wybranych atrybutów systemu plików ext4

Jak wspomniano, istnieje możliwość zmiany wartości niektórych atrybutów systemu plików z rodziny ext. Służy do tego, niezależnie od wersji systemu plików, program *tune2fs*. Jak widzieliśmy, jego możliwości są duże, gdyż pozwala on nawet na konwersję systemu plików ext z wersji 3 do 4. Użycie go w minimalnym zakresie polega na wypisaniu wartości atrybutów, co wymaga użycia opcji **-l** oraz podania, jako argumentu, nazwy pliku reprezentującego partycję, na której założono system plików. Przykładowy wynik działania przedstawiono poniżej:

```

1 [root@messy ~]# tune2fs -l /dev/sda7
2 tune2fs 1.41.9 (22-Aug-2009)
3 Filesystem volume name:   <none>
4 Last mounted on:         <not available>
5 Filesystem UUID:         cf733052-85a0-42e5-984c-e107980ed407
6 Filesystem magic number:  0xEF53
7 Filesystem revision #:    1 (dynamic)
8 Filesystem features:      has_journal ext_attr resize_inode dir_index filetype extent flex_bg
9 Filesystem flags:         signed_directory_hash
10 Default mount options:    (none)
11 Filesystem state:         clean
12 Errors behavior:          Continue
13 Filesystem OS type:       Linux
14 Inode count:              33200
15 Block count:              132528
16 Reserved block count:     6626
17 Free blocks:              126239
18 Free inodes:              33189
19 First block:              0

```

```

20 Block size:                4096
21 Fragment size:           4096
22 Reserved GDT blocks:     32
23 Blocks per group:        32768
24 Fragments per group:     32768
25 Inodes per group:        6640
26 Inode blocks per group:  415
27 Flex block group size:   16
28 Filesystem created:      Wed Oct 26 13:35:57 2011
29 Last mount time:         n/a
30 Last write time:         Wed Oct 26 13:35:59 2011
31 Mount count:             0
32 Maximum mount count:     25
33 Last checked:            Wed Oct 26 13:35:57 2011
34 Check interval:          15552000 (6 months)
35 Next check after:        Mon Apr 23 13:35:57 2012
36 Lifetime writes:         24 MB
37 Reserved blocks uid:     0 (user root)
38 Reserved blocks gid:     0 (group root)
39 First inode:             11
40 Inode size:              256
41 Required extra isize:    28
42 Desired extra isize:     28
43 Journal inode:           8
44 Default directory hash:  half_md4
45 Directory Hash Seed:     66d63fa6-3048-460c-ba16-f63405662d21
46 Journal backup:          inode blocks

```

Najistotniejsze informacje to:

- linia 3 – system nie posiada etykiety. Można ją ustalić programem *e2label*.
- linia 4 – system plików nie jest zamontowany.
- linia 5 – definiuje unikalny identyfikator urządzenia, który jest obecnie wykorzystywany do jego identyfikacji w pliku */etc/fstab*.
- linia 8 – to lista określająca cechy systemu plików.
- linia 11 – zawiera informacje o stanie spójności systemu.
- linia 15 – to liczba bloków danych dostępnych w systemie plików.
- linia 16 – liczba bloków zarezerwowanych, do wykorzystania przez użytkownika *root*.
- linia 17 – liczba bloków dostępnych.
- linia 18 – liczba dostępnych i-węzłów.
- linia 19 – to numer pierwszego dostępnego bloku danych.
- linia 20 – rozmiar bloku danych w bajtach.

- linia 21 – rozmiar fragmentu w bajtach.
- linia 23 – to liczba bloków przypadająca na jedną grupę systemu plików.
- linia 24 – to liczba fragmentów przypadających na jedną grupę.
- linia 25 – liczba i-węzłów znajdujących się w jednej grupie.
- linia 28 – zawiera datę utworzenia systemu plików.
- linia 29 – to data ostatniego montowania systemu plików. Przykładowy system plików po utworzeniu nie był montowany.
- linia 30 – to data ostatniego zapisu dokonanego w systemie plików (danych lub metadanych).
- linia 31 – zawiera licznik montowań systemu plików.
- linia 32 – wartość ta oznacza liczbę montowań, przekroczenie której spowoduje przy najbliższym montowaniu systemu plików sprawdzenie jego spójności programem *fsck*.
- linia 33 – to informacja o dacie, kiedy ostatni raz sprawdzano spójność systemu plików.
- linia 34 – minimalna liczba sekund pomiędzy kolejnymi badaniami spójności systemu plików programem *fsck* (przyjmuje się, że miesiąc ma 30 dni).
- linia 35 – to data, po której montowanie systemu plików zostanie poprzedzone badaniem spójności systemu plików programem *fsck*.
- linia 39 – zawiera numer pierwszego wolnego i-węzła z listy wolnych i-węzłów.
- linia 43 – to numer i-węzła opisującego plik dziennika, który w tym przypadku został założony w postaci pliku znajdującego się w danym systemie plików.

#### Montowanie systemu plików ext4

Dla zamontowania systemu plików ext4 można skorzystać z następujących opcji:

- **extents** – opcja ta spowoduje, że system plików będzie przydzielał bloki danych w grupach. Pierwsze zamontowanie systemu plików z tą opcją spowoduje, że nie będzie można go później zamontować jako system ext3.
- **journal=update** – opcja umożliwia zmianę formatu dziennika systemu plików do aktualnego.
- **journal=inum** – jeśli montowany system plików posiada dziennik, opcja jest ignorowana. W przeciwnym przypadku jej wartość oznacza liczbę i-węzłów utworzonych w dzienniku montowanego systemu plików.
- **journal\_dev=devnum** – opcja jest wykorzystywana w przypadku przeniesienia dziennika na inne urządzenie. Jej wartość pozwala określić numery *major* oraz *minor* urządzenia, na którym dziennik obecnie się znajduje.
- **noload** – użycie opcji spowoduje, że zamontowanie systemu plików odbędzie się bez sięgania do dziennika czyli odnotowania zapisanych w nim tranzakcji.



- **data=journal** – opcja determinuje sposób działania mechanizmu dziennikowania. W zamontowanym z jej wykorzystaniem systemie plików zmiany wszystkich danych zostaną zapisane najpierw w dzienniku, a następnie odnotowane w systemie plików.
- **data=ordered** – w tym trybie pracy mechanizmu dziennikowania dane zostaną zapisane bezpośrednio do systemu plików, zaś zmiany w metadanych zostaną odnotowane w dzienniku. Opcja z tą wartością jest wykorzystywana domyślnie.
- **data=writeback** – ten tryb pracy mechanizmu dziennikowania polega na odnotowaniu w systemie plików zmian tych danych, których zmiany metadanych zostały odnotowane w dzienniku.
- **commit=liczba\_sekund** – opcja umożliwia wykorzystanie właściwości systemu plików umożliwiającej synchronizację danych oraz metadanych (odnotowywanie zmian danych oraz metadanych zapisanych w dzienniku w systemie plików) co liczbę sekund będącą wartością opcji. Wartość domyślna to 5 sekund. Wartość mniejsza spowoduje zmniejszenie wydajności systemu, ale wpłynie korzystnie na bezpieczeństwo danych przechowywanych w systemie plików. Użycie opcji z wartością 0 spowoduje nadanie wartości domyślnej. Stosowanie dużych wartości powoduje zwiększenie wydajności systemu.
- **barrier=1** –
- **orlov** – użycie opcji powoduje, że do rozmieszczania katalogów systemu plików na urządzeniu zostanie użyty algorytm Orłowa. W dużym skrócie, algorytm ten próbuje umieścić kilka katalogów oraz opisujące je i-węzły w jednej grupie cylindrów. Dodatkowo rezerwowane jest miejsce na i-węzły opisujące pliki, które w przyszłości znajdą się w katalogu oraz na bloki danych przechowujących dane zapisane w tych plikach. Pewną wadą algorytmu jest to, iż powoduje on relatywnie dużą defragmentację plików. Algorytm jest stosowany w systemie plików FFS. W ext4 jest on domyślny.
- **oldalloc** – w zamontowanym z tą opcją systemie plików do rozmieszczania katalogów oraz i-węzłów i bloków danych znajdujących się w nich plików na cylindrach i grupach cylindrów zostaną użyte tradycyjne algorytmy. Ich działanie jest bardzo proste i polega na znalezieniu grupy cylindrów, w obrębie której zapisanych jest najmniejsza liczba katalogów. Nowy katalog jest zazwyczaj umieszczany w innej grupie cylindrów niż jego katalog macierzysty, co negatywnie wpływa na wydajność operacji dostępu do zapisanych w nim danych. Staje się to wyraźnie widoczne zwłaszcza w dużych systemach plików.
- **user\_xattr** – opcja umożliwia definiowanie przez użytkowników dodatkowych atrybutów plików. Wymaga to jednak ustawienia atrybutu `CONFIG_EXT4_FS_XATTR` podczas kompilacji kodu jądra obsługującego system plików ext4. Każdy atrybut to para nazwa-wartość. Do ich obsługi służy polecenie `attr`.
- **nouser\_xattr** – wyłącza możliwość wprowadzania dodatkowych atrybutów użytkownika.
- **acl** – użycie opcji jest konieczne jeśli dla opisu praw dostępu do plików znajdujących się w montowanym systemie plików mają być wykorzystywane listy kontroli dostępu. Dodatkowo, wymaga się ustawienia atrybutu `CONFIG_EXT4_FS_POSIX_ACL` podczas kompilacji kodu jądra mającego obsługiwać system plików ext4.
- **noacl** – opcja wyłącza możliwość stosowania w systemie plików list kontroli dostępu.

- **reservation** – opcja wymusza, aby dla nowoutworzonego katalogu zarezerwować pewną ilość miejsca na i-węzły plików, które zostaną w nim utworzone. Jej efektywne działanie zależy od użytego algorytmu rezerwacji miejsca dla tworzonego katalogu.
- **noreservation** – użycie opcji spowoduje, że wraz z utworzeniem nowego katalogu nie będzie rezerwowane miejsce dla i-węzłów przechowywanych w nim plików.
- **bsddf** – opcja ustawia tryb działania komendy *df* jak w systemie BSD. Tryb ten jest domyślnym.
- **minixdf** – tryb pracy komendy *df* jest jak w systemie Minix.
- **check=none** – opcja przyspieszy proces montowania systemu plików, gdyż spowoduje, że nie będzie sprawdzana poprawność map bitowych.
- **debug** – użycie opcji spowoduje, że do dziennika systemowego będą generowane dodatkowe informacje o pracy systemu plików.
- **errors=remount-ro** – jeśli podczas pracy systemu plików wystąpi błąd, to dzięki użyciu tej opcji zostanie on ponownie zamontowany w trybie tylko do odczytu.
- **errors=continue** – użycie tej opcji spowoduje brak reakcji w przypadku wystąpienia błędu w pracy systemu plików.
- **errors=panic** – opcja ta powoduje najostrzejszą reakcję na wystąpienie błędu podczas pracy systemu plików, sprowadzającą się do zatrzymania pracy systemu operacyjnego.
- **grpuid|bsdgroups** – wybranie tej opcji spowoduje, że właścicielem grupowym każdego utworzonego w systemie plików obiektu będzie właściciel grupowy katalogu, w którym obiekt ten zostanie utworzony.
- **nogrpuid|sysvgroups** – w wyniku działania opcji, właścicielem grupowym każdego utworzonego w systemie plików obiektu będzie grupa podstawowa użytkownika będącego właścicielem indywidualnym procesu, który obiekt ten utworzył. Ustawienie to jest domyślne.
- **resgid=n** – opcja pozwala określić numer identyfikacyjny grupy użytkowników, która będzie uprawniona do korzystania z zarezerwowanych bloków danych. Grupą domyślną jest grupa o numerze 0.
- **resuid=n** – wartość opcji to numer identyfikacyjny użytkownika, który będzie uprawniony do wykorzystywania zarezerwowanych bloków danych. Domyślnie jest to użytkownik o numerze 0.
- **sb=n** – wartość opcji pozwala określić numer bloku identyfikacyjnego, który będzie traktowany jako podstawowy.
- **quota, noquota, grpquota, usrquota** – użycie tych opcji jest wymagane, jeśli w systemie plików zamierzamy reglamentować i-węzły oraz bloki dane zajęte przez użytkownika lub grupę użytkowników.
- **bh|nobh** – system plików ext4 wykorzystuje powiązania nagłówków buforów dyskowych ze stronami danych w celu przyspieszenia działania pamięci podręcznej oraz oznaczania stron należących do danej tranzakcji w celu zapewnienia właściwej kolejności ich obsługi. Opcja **bh** wymusza użycie nagłówków buforów (ustawienie domyślne). Opcja **nobh** powoduje, że nagłówki buforów dyskowych nie będą wykorzystywane (działa efektywnie w trybie dziennikowania metadanych).

**Badanie spójności systemu plików ext4**

Do badania spójności systemu plików ext4 służy program *fsck.ext4*. W przypadku najprostszego uruchomienia wymaga podania nazwy pliku reprezentującego urządzenie, na którym znajduje się badany system plików. Zalecane jest, aby system plików nie był zamontowany, gdyż badanie spójności może doprowadzić do jego uszkodzenia. Wyjątek stanowi użycie opcji **-n** oraz nie wyspecyfikowanie opcji **-c**, **-l** i **-L** podczas uruchamiania programu.

```

1 [root@messy ~]# fsck.ext4 -n /dev/sda7
2 e2fsck 1.41.9 (22-Aug-2009)
3 Warning! /dev/sda7 is mounted.
4 Warning: skipping journal recovery because doing a read-only filesystem check.
5 ext4_on_sda7: clean, 2575/33200 files, 10664/132528 blocks

```

Opcja **-n** gwarantuje dostęp do systemu plików w trybie tylko do odczytu oraz zapewnia niejawnie udzielanie odpowiedzi przeczącej na każde pytanie zadane przez program *fsck.ext4*. Stąd badanie spójności systemu plików w tym przypadku przebiegło bez sięgnięcia do danych zapisanych w dzienniku. Pominięcie opcji podczas weryfikacji spójności zamontowanego systemu plików spowoduje, że program będzie się domagał potwierdzenia kontynuacji. Następnie zostaną odnotowane w systemie wszystkie operacje znajdujące się w dzienniku i tak uaktualniony system plików zostanie zweryfikowany.

```

1 [root@messy scratch1]# fsck.ext4 /dev/sda7
2 e2fsck 1.41.9 (22-Aug-2009)
3 /dev/sda7 is mounted.
4
5 WARNING!!! Running e2fsck on a mounted filesystem may cause
6 SEVERE filesystem damage.
7
8 Do you really want to continue (y/n)? yes
9
10 ext4_on_sda7: recovering journal
11 ext4_on_sda7: clean, 2575/33200 files, 10217/132528 blocks

```

Skuteczność programu *fsck.ext4* zbadamy uszkadzając pierwszy fragment systemu plików zapisując 64 kB danych o wartościach losowych pochodzących z urządzenia */dev/urandom*.

```

1 [root@messy ~]# fsck.ext4 /dev/sda7
2 e2fsck 1.41.9 (22-Aug-2009)
3 fsck.ext4: Superblock invalid, trying backup blocks...
4 One or more block group descriptor checksums are invalid. Fix<y>? yes
5
6 Group descriptor 0 checksum is invalid. FIXED.
7 Group descriptor 1 checksum is invalid. FIXED.
8 Group descriptor 2 checksum is invalid. FIXED.
9 Group descriptor 3 checksum is invalid. FIXED.
10 Group descriptor 4 checksum is invalid. FIXED.
11 ext4_on_sda7 contains a file system with errors, check forced.
12 Resize inode not valid. Recreate<y>? yes

```

```

13
14 Pass 1: Checking inodes, blocks, and sizes
15 Pass 2: Checking directory structure
16 Pass 3: Checking directory connectivity
17 Pass 4: Checking reference counts
18 Pass 5: Checking group summary information
19 Free blocks count wrong for group #0 (30642, counted=30529).
20 Fix<y>? yes
21
22 Free blocks count wrong for group #1 (32734, counted=28473).
23 Fix<y>? yes
24 .....
25
26 Free inodes count wrong (33189, counted=30625).
27 Fix<y>? yes
28
29
30 ext4_on_sda7: ***** FILE SYSTEM WAS MODIFIED *****
31 ext4_on_sda7: 2575/33200 files (0.0% non-contiguous), 10664/132528 blocks
32 [root@messy ~]# fsck.ext4 /dev/sda7
33 e2fsck 1.41.9 (22-Aug-2009)
34 ext4_on_sda7: clean, 2575/33200 files, 10664/132528 blocks
35 [root@messy ~]# mount -t ext4 /dev/sda7 /scratch1

```

Jak widać strategia działania programu dla systemu plików ext4 jest analogiczna jak w przypadku jego odpowiednika dla systemu plików ext3. Po stwierdzeniu nieprawidłowości występujących w bloku identyfikacyjnym (linia 3) zweryfikowane zostały sumy kontrolne w kolejnych grupach deskryptorów systemu plików (linie 4–10). Przywracanie spójności systemu plików przebiegało w pięciu fazach, które obejmowały: 1. weryfikację poprawności informacji zapisanej w i-węzłach (w tym tablicy alokacji bloków danych), 2. kontrolę poprawności wpisów katalogowych (nazwa, długość), 3. analizę struktury drzewa katalogów (czy do każdego katalogu prowadzi jedna ścieżka), 4. sprawdzenie poprawności liczników dowiązań wszystkich i-węzłów, 5. Weryfikację struktur opisujących grupy, przeprowadzoną z wykorzystaniem informacji zebranych w poprzednich fazach. W naszym przypadku pojawiły się niepoprawne wartości opisujące listy wolnych bloków danych oraz i-węzłów (liczniki), które zostały skorygowane. Powtórne uruchomienie programu *fsck.ext4* pozwoliło stwierdzić, że system plików jest spójny, a poprawność bloku identyfikacyjnego potwierdzono montując system plików. Pozwoliło to odzyskać dane, które zostały zapisane w nieuszkodzonych blokach danych.

Na zakończenie zwróćmy uwagę na kilka naczęściej wykorzystywanych opcji. Należą do nich:

- **-b numer\_bloku\_identyfikacyjnego** – opcja umożliwia wskazania programowi numeru bloku identyfikacyjnego z którego ma korzystać. Jest wykorzystywana w przypadku uszkodzenia głównego bloku identyfikacyjnego.
- **-c** – opcja powoduje uruchomienie programu *badblocks* w celu uczynienia system plików tylko do odczytu oraz znalezienia bloków uszkodzonych. Użycie dwukrotne opcji powoduje, iż poszukiwanie bloków uszkodzonych jest przeprowadzane nie niszczącym zawartości bloków testem odczytu i zapisu.
- **-l nazwa\_pliku** – użycie opcji powoduje dodanie do listy bloków uszkodzonych, bloków,

których numery zostały umieszczone w pliku o nazwie będącej argumentem opcji. Format pliku musi być zgodny z wykorzystywanym przez program *badblocks*.

- **-I nazwa\_pliku** – działanie opcji polega na ustawieniu zawartości listy bloków uszkodzonych na podstawie zawartości pliku, którego nazwa jest argumentem opcji. Różnicą w stosunku do działania opcji **-l** jest usuwanie poprzedniej zawartości listy bloków uszkodzonych, zanim zostaną dodane do niej te, znajdujące się w pliku.
- **-j urządzenie\_przechowujące\_dziennik** – opcja jest konieczna jeśli dokonujemy weryfikacji systemu plików, który posiada dziennik na innym urządzeniu. Wartością opcji jest ścieżka dostępu do pliku reprezentującego urządzenie.
- **-D** – użycie opcji wymusza, aby program *fsck.ext4* dokonał optymalizacji zawartości wszystkich znajdujących się w systemie plików katalogów.
- **-f** – opcja wymusza sprawdzanie spójności systemu plików nawet jeśli jest on zaznaczony jako spójny.
- **-p** – powoduje podjęcie próby automatycznej naprawy znalezionych niespójności bez interwencji użytkownika. Jednocześnie raportuje znalezione nieprawidłowości.

## Programy narzędziowe

### 3.4.5 System plików XFS

System plików XFS został opracowany przez firmę Silicon Graphics Inc. (SGI) dla systemu operacyjnego IRIX. Jego premiera miała miejsce w roku 1994. System bazuje koncepcyjnie na omówionym systemie FFS, jednak z pewnymi modyfikacjami, do których zaliczyć należy wyeliminowanie wstępnej alokacji i-węzłów oraz wyposażenie go w kronikę. W roku 2000 został on udostępniony jako oprogramowanie *open source*. Stąd pojawił się w systemach linuksowych jak i innych, należących do rodziny systemów uniksowych. Duża liczba użytkowników i implementacji dla różnych systemów komputerowych sprawiły, że system ten został wszechstronnie przetestowany i zalicza się do bardzo stabilnych. Do cech charakterystycznych zaliczyć należy:

- Możliwość obsługi plików o dużych rozmiarach. Dodatkowo możliwość przechowywania dużej liczby plików czyli tworzenia katalogów o dużych rozmiarach, a w konsekwencji przechowywanie olbrzymich ilości danych. Cechy te wynikają z przeznaczenia systemu operacyjnego IRIX m.in. dla stacji roboczych przystosowanych do obróbki grafiki dużej rozdzielczości i multimediiów. System plików zapewnia wysoką przepustowość strumieni audio i wideo.
- Obsługę plików udostępnianych w czasie rzeczywistym. Możliwość tę uzyskano dzięki umieszczeniu plików w specjalnym obszarze systemu plików.
- Wykorzystanie zaawansowanych algorytmów wyszukiwania plików i katalogów oraz alokacji przestrzeni dyskowej. Konsekwencją jest skrócenie czasu dostępu do informacji przechowywanej w systemie plików.

## Budowa wewnętrzna

Struktura wewnętrzna systemu plików XFS to trzy podstawowe obszary:

1. Obszar danych, w którym przechowywane są dane z plików oraz metadane systemu plików.

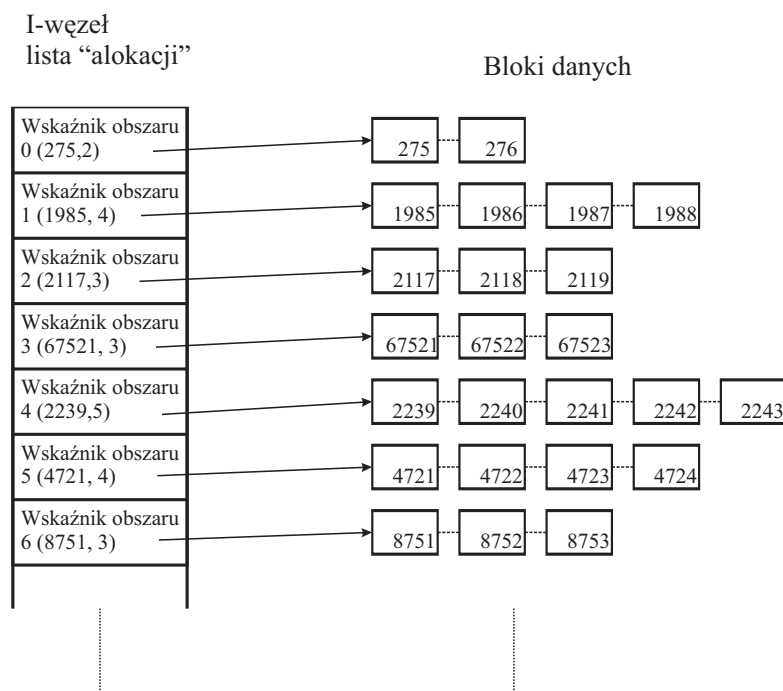
2. Dziennika, w którym rejestrowane są transakcje opisujące zmiany metadanych systemu plików.
3. Sekcja czasu rzeczywistego (opcjonalna), która służy do przechowywania danych z plików wymagających ciągłych, szybkich operacji wejścia/wyjścia.

Obszar danych został podzielony na grupy alokacji, koncepcyjnie odpowiadającym grupom cylindrów z systemu FFS. Grupy są zarządzane przez system plików bez możliwości ingerencji ze strony administratora systemu. Ich rozmiar waha się od 0.5 do 4 GB. Każda grupa alokacji posiada własne struktury danych zarządzające i-węzłami oraz blokami danych w ramach grupy. Podejście takie ma szereg zalet, do których należą:

- Zwiększenie wydajności możliwe dzięki redukcji opóźnień blokad i umożliwienie współbieżnej (lub równoległej) obsługi przydzielania i dostępu do bloków danych i i-węzłów w różnych grupach alokacji.
- Uproszczenie mechanizmu zmiany rozmiarów plików. W systemie plików wykorzystującym grupy alokacji zmiany te polegają na dodaniu nowych lub usunięciu istniejących grup.
- Możliwość wykorzystania w obrębie grupy alokacji względnych wskaźników do i-węzłów oraz bloków danych, co pozwala zaoszczędzić miejsce w wewnętrznych strukturach danych grup alokacji oraz zwiększyć potencjalny rozmiar systemu plików XFS w porównaniu do systemów plików pracujących na wskaźnikach jawnych i bezwzględnych.

Podstawowym założeniem techniki grup cylindrów było przydzielanie bloków danych dla danych należących do tego samego pliku w tej samej grupie cylindrów dyskowych. Jak wiemy, mechanizm ten pozwala zmniejszyć narzuty czasowe związane z koniecznością przesuwania głowicy dysku podczas dostępu do pliku. Niestety stosowany w systemie plików XFS rozmiar grupy alokacji zmniejsza prawdopodobieństwo umieszczenia danych pliku w kolejnych blokach. W zamian wprowadzono pojęcie *skupienia logicznego*, polegającego na optymalnym w ramach grupy alokacji rozmieszczeniu wszystkich obszarów dysku przechowujących dane należące do danego pliku. Realizacja praktyczna znalazła swoje odzwierciedlenie między innymi w strukturach danych przechowujących informacje o rozmieszczeniu danych pliku w blokach danych. Rozwiązanie klasyczne przedstawiono na rysunku 3.6. W i-węźle znajduje się tablica przechowująca numery bloków zawierających bezpośrednio dane pliku lub numery bloków danych z tablicami adresów pośrednich. Każda pozycja w tablicy zawiera jednak numer pojedynczego bloku danych. Zwróćmy uwagę, iż numery bloków podane przykładowo po stronie prawej rysunku różnią się znacznie. Bloki te leżą zatem w różnych fragmentach systemu plików, a w konsekwencji w różnych obszarach dysku. W takiej sytuacji sekwencyjny odczyt danych z pliku będzie wymagał częstych ruchów głowicą dysku, co automatycznie zmniejszy wydajność operacji. System plików XFS w miejsce alokacji pojedynczych bloków stosuje model oparty na obszarach. Jak przedstawiono na rysunku 3.14, obszary są ciągami przylegających do siebie bloków dyskowych, alokowanych wspólnie. Adresujące je struktury danych przechowują numer pierwszego bloku obszaru oraz jego długość.

Zastosowanie modelu alokacji obszarów pozwala osiągnąć większą wydajność operacji wejścia/wyjścia w odniesieniu do danych przechowywanych w plikach jak również większą wydajność operacji przydziału bloku danych. W stosunku do mechanizmu stosowanego w Unix System V pozwala również na redukcję rozmiaru struktur metadanych. Pewna niedogodność alokowania przylegających do siebie bloków danych pojawia się w momencie awarii dysku. Rośnie mianowicie prawdopodobieństwo utraty wszystkich danych zapisanych w pliku, jeśli zajmowały one w uszkodzonym obszarze. W przypadku zapisu danych w pojedynczych, rozproszonych na większym obszarze dysku blokach prawdopodobieństwo utraty całości jest mniejsze.



Rysunek 3.14: System plików XFS musi alokować kolejne bloki danych dla przechowywania danych pliku. Stąd w i-węźle pamiętana jest lista zaalokowanych ciągów bloków, której elementy zawierają numer początkowego bloku danych oraz ich liczbę w ciągu.

Zwiększeniu wydajności systemu ma służyć również zastosowanie pewnych heurystyk. Przykładowo, każdy tworzony katalog jest umieszczany w innej grupie alokacji niż jego katalog nadrzędny, a dla przyszłych jego elementów rezerwowana jest pewna ilość ciągłej przestrzeni dyskowej. System próbuje również alokować i-węzły oraz obszary bloków danych przez nie opisywane w bezpośrednim sąsiedztwie obszaru dysku, w którym znajdują się struktury opisujące zawierające katalog.

I-węzeł w systemie plików XFS, podobnie jak w innych poznanych dotychczas systemach, jest podstawową strukturą opisującą plik. Jednakże system ten alokuje i-węzły w sposób dynamiczny co oznacza, że lista i-węzłów nie jest inicjalizowana w momencie tworzenia systemu plików, a powstaje dynamicznie w trakcie eksploatacji systemu plików. Podejście takie pozwala z jednej strony na racjonalną gospodarkę przestrzenią adresową dostępną w systemie plików z drugiej zwiększa jego elastyczność nie wymuszając zdefiniowania rozmiaru listy i-węzłów już w momencie zakładania systemu plików. Praktycznie system ten może zawierać dowolną liczbę plików i katalogów, ograniczoną jedynie ilością dostępnego miejsca. Stąd zakładając go nie musimy zastanawiać się jak dopasować długość listy i-węzłów do liczby bloków danych, gdyż proporcja ta będzie ustalana na bieżąco, podczas pracy systemu.

Interesującą cechą systemu plików XFS jest obsługa plików rzadkich (ang. *sparse files*). Są to pliki o dużych rozmiarach, których nieliczne bajty mają wartość różną od 0. Przechowywanie danych w przypadku takich plików można ograniczyć do zapamiętania wartości bajtów różnych od zera oraz liczby rozdzielających je bajtów zerowych. Choć rozmiar plików rzadkich zwracanych przez metadane systemu plików jest jego rozmiarem maksymalnym, to w praktyce zajmuje on znacznie mniej przestrzeni adresowej.

Na wydajność systemu plików XFS ma wpływ nie tylko sposób rozmieszczenia danych, ale również przechowywanie metadanych w węzłach zrównoważonych drzew binarnych B+. B+ drzewa wykorzystywane są do:

- zarządzania dynamicznie alokowanymi i-węzłami. W obrębie grupy alokacji tworzone są 64 i-węzły, a B+ drzewo umożliwia szybkie odnalezienie wolnych oraz opisujących konkretne pliki.
- Szybkiego odnajdywania wolnych obszarów bloków danych. Zastosowanie drugiego indeksowania po rozmiarze obszaru umożliwia optymalne zarządzanie wolną przestrzenią danych w systemie plików. Pozwala także na efektywniejsze wykorzystywanie adresów bezpośrednich w i-węźle pliku, co przekłada się bezpośrednio na szybkość operacji wejścia-wyjścia.
- Przyspieszania wyszukiwania zawartości katalogów. Umożliwia to mechanizm indeksowania nazw katalogów wykorzystujący ich cztero bajtowe skróty uporządkowane w strukturę drzewiastą.

Interesującą cechą systemu plików XFS jest możliwość tworzenia podwolumenów czasu rzeczywistego, zapewniających utrzymania gwarantowanej przepustowości dla operacji wejścia-wyjścia (ang. *Guaranteed Ratio I/O, GRIO*). Początkowo były one wykorzystywane do przechowywania plików graficznych. Obecnie nie trudno znaleźć dla nich inne zastosowania, jak chociażby przechowywanie plików multimedialnych. Techniczna realizacja podwolumenów polega na przechowywaniu opisujących je metadanych oraz metadanych znajdujących się w nich plików w głównej części systemu plików. Dodatkowo wykorzystywany algorytm przydziału bloków danych został zoptymalizowany pod kątem minimalnego czasu alokacji oraz minimalnej fragmentacji. Wskazanie, że nowy plik tworzony w systemie plików XFS wymaga operacji czasu rzeczywistego, odbywa się za pomocą funkcji *ioctl*. Istniejące pliki można skopiować do podwolumenu czasu rzeczywistego korzystając z programu narzędziowego *xfstcp*, który nada im automatycznie odpowiednie atrybuty.

### Zakładanie systemu plików XFS

Oprócz dodania do jądra systemu operacyjnego funkcji służących do obsługi systemu plików XFS, konieczne jest zainstalowanie programów narzędziowych umożliwiających jego zakładanie, montowanie oraz obsługę. Dostępne są one w pakiecie *xfstools*, który zawiera ich kod źródłowy. Jest on do znalezienia w postaci skompresowanego archiwum, np. na stronie <http://linux.softpedia.com>. Dodatkowe programy narzędziowe dla systemu plików XFS można znaleźć na stronie <http://xfs.org>.

Podobnie jak w przypadku innych systemów plików dostępnych w systemach linuksowych, system plików XFS zakłada się przy pomocy programu *mkfs* z katalogu */sbin* uruchomionego z opcją **-t** i wartością *xfs*. Ostatecznie zostaje uruchomiony program *mkfs.xfs*, do którego zostają przekazane wszystkie opcje i argumenty programu *mkfs*. W najprostszym przypadku, oprócz wspomnianej opcji **-t** wymagane jest podanie argumentu, którym jest ścieżka dostępu do pliku urządzenia. Jeśli na wskazanym urządzeniu znajduje się system plików, to wówczas należy użyć opcji **-f**:

```

1 [root@messy ~]# mkfs -t xfs -f /dev/sda7
2 meta-data=/dev/sda7          isize=256      agcount=4, agsize=33132 blks
3      =                      sectsz=512      attr=2
4 data      =                  bsize=4096     blocks=132528, imaxpct=25
5      =                      sunit=0         swidth=0 blks
6 naming    =version 2         bsize=4096     ascii-ci=0

```



7	log	=internal log	bsize=4096	blocks=1200, version=2
8		=	sectsz=512	sunit=0 blks, lazy-count=0
9	realtime	=none	extsz=4096	blocks=0, rtextents=0

Zakładanie systemu plików XFS trwa bardzo szybko w porównaniu do systemu plików ext3. To m.in. dlatego, że nie są wstępnie zakładane niektóre struktury systemu plików, jak np. i-węzły. Informacje wypisywane podczas tworzenia systemu plików dotyczą jego poszczególnych sekcji. Sekcja *meta-data* zawiera wartości atrybutów metadanych systemu plików:

- *isize* – rozmiar i-węzła w bajtach. Wartością domyślną jest 256 bajtów.
- *agcount* – liczba grup alokacji. Zależy ona od rozmiaru systemu plików.
- *agsize* – rozmiar grupy alokacji w blokach danych. Jest on obliczany na podstawie rozmiaru partycji, na której zakładany jest system plików oraz liczby grup alokacji. Przyjmuje się, że powinien on być z zakresu od 16 MB do 4 GB.
- *sectsz* – rozmiar podstawowego sektora danych w bajtach. Wartość domyślna i minimalna zarazem to 512, zaś wartość maksymalna 32768 bajty.
- *attr* – numer wersji polityki alokowania atrybutów rozszerzonych. Obecnie, od jądra systemu operacyjnego 2.6.16 jest ona domyślną. Wykorzystuje ona efektywny algorytm zarządzający dostępnym w i-węźle miejscem dzieląc go dynamicznie między wartości atrybutów, a wpisy. Wersja 1, dostępna również dla starszych wersji jądra, zapewniała stały podział miejsca, co prowadziło do jego marnotrawienia.

Wartości atrybutów danych opisuje sekcja *data*:

- *bsize* – rozmiar bloku danych w bajtach. Wartość domyślna to 4096 B. Może przyjmować wartości z przedziału od 512 do 65536 B.
- *blocks* – całkowita liczba bloków danych dostępnych w systemie plików. Jest ona obliczana na podstawie rozmiaru bloku oraz rozmiaru partycji, na której zakładany jest dany system plików, przy czym rozmiar ten jest pomniejszany o rozmiar zajmowany przez dziennik, rozmiar sekcji czasu rzeczywistego oraz miejsce zajmowane przez struktury opisujące grupy alokacji.
- *imaxpct* – określa maksymalny obszar systemu plików zajmowany przez i-węzły, podawany w procentach. Wartość 25 jest domyślna.
- *sunit* – rozmiar jednostki przepływu (*stripe unit*) podawana w liczbie bloków danych o rozmiarze 512 B (problematyka macierzy dyskowych została poruszona w podrozdziale 3.7). Domyślnie przepływ jest wyłączony, stąd wartość 0.
- *swidth* – rozmiar wstęgi przepływu (*stripe width*) podawana w liczbie bloków danych o rozmiarze 512 B.

Atrybuty katalogów znajdują się w sekcji *naming*:

- *version* – określa typ katalogów obsługiwanych przez system XFS. Typ 2 jest domyślnym i zalecanym. Typ 1 nie jest obsługiwany.
- *bsize* – określa rozmiar bloku katalogu w bajtach. Wartość 4096 jest domyślna. Dla wersji 2 może przyjmować wartość od rozmiaru pojedynczego bloku danych do wartości 65536.

- *ascii-ci* – pozwala na rozróżnianie lub nie dużych i małych liter w nazwach katalogów.

Właściwości dziennika zawiera sekcja *log*:

- *internal log* – dziennik systemu plików XFS może być przechowywane w jego obszarze lub na innym urządzeniu.
- *bsize* – rozmiar bloku dziennika podawany w bajtach. Wartość domyślna to 4096.
- *blocks* – liczba bloków tworząca dziennik.
- *version* – wersja dziennika. Wersja 2 pozwala stosować większe rozmiary bufora zapisu oraz paskowanie (RAID 0). Dostępna dla jąder systemu operacyjnego 2.6. Wersja 1 posiada ograniczenie bufora zapisu do 32 kB i jest obsługiwana przez jądra w wersji 2.4.
- *sectsz* – rozmiar sektora dziennika podawany w bajtach. Rozmiar domyślny to 512 B.
- *sunit* – wartość określa rozmiar porcji zapisywanych do dziennika. Podawany jest w blokach o rozmiarze 512 B.
- *lazy-count* – określa sposób zapisywania wartości różnych liczników w bloku identyfikacyjnym systemu plików. Wartość 1 oznacza, że wartości przechowywane w bloku identyfikacyjnym nie będą uaktualniane przy każdej ich zmianie. W innych zmiennych systemu plików będzie natomiast przechowywana informacja, która w każdym momencie pozwoli na wyznaczenie ich wartości w celu uaktualnienia zawartości bloku identyfikacyjnego. Pozwala to zwiększyć wydajność systemu plików. Wartością domyślną jest 0 (off). Wprowadzenie wartości zmiennych charakteryzujących stan systemu plików zapisane w bloku identyfikacyjnym będą uaktualniane na bieżąco, ale obniży to wydajność systemu plików.

Obszar czasu rzeczywistego opisuje sekcja *real-time*:

- *extsz* – określa rozmiar bloku danych w sekcji czasu rzeczywistego. Wartość minimalna jest równa rozmiarowi bloku danych w systemie plików lub 4 kB, w zależności co jest większe. Wartość domyślna jest równa rozmiarowi paska w przypadku stosowania paskowania (RAID 0) lub 64 kB dla wolumenów, dla których technika ta nie jest stosowana. Wartość maksymalna nie może przekraczać 1 GB.
- *blocks* – wartość oznacza całkowitą liczbę bloków danych sekcji czasu rzeczywistego. Jeśli sekcja czasu rzeczywistego nie została utworzona wartość wynosi 0.
- *rtextents* – określa liczbę sekcji czasu rzeczywistego utworzonych w danym systemie plików. Wartość domyślna to 0 (program *mkfs.xfs* domyślnie nie tworzy sekcji czasu rzeczywistego).

Wartości wszystkich wymienionych atrybutów systemu plików XFS mogą zostać nadane w zależności od potrzeb w trakcie tworzenia systemu plików. W następnym punkcie przedstawiono przykłady dla niektórych z nich.

### Modyfikacja wartości wybranych atrybutów systemu plików XFS

Jednym z najczęściej modyfikowanych atrybutów tworzonego systemu plików XFS jest rozmiar dziennika. Określenie rozmiaru wymaga użycia opcji **-l** ze specyfikatorem *size=* i wartością będącą rozmiarem dziennika. Rozmiar może zostać podany w bajtach i wówczas po wartości nie podajemy jednostki. Rozmiar podany w blokach określa litera *b*, w kilobajtach litera *k*, w megabajtach litera *m*, w gigabajtach litera *g*.

Jeden ze sposobów zwiększenia wydajności dziennikowanego systemu plików polega na umieszczeniu dziennika na innym urządzeniu. Podobnie jak zmiana innych atrybutów dziennika wymagane jest użycie opcji **-l** ze specyfikatorem *logdev*, którego wartością jest ścieżka dostępu do pliku urządzenia, na którym ma zostać założony dziennik.

Przykładowo, na urządzeniu */dev/sda7* utworzymy system plików XFS z dziennikiem o rozmiarze 10 GB położonym na urządzeniu */dev/sdc1*. Oczywiście wymagane jest wcześniejsze utworzenie partycji na dysku */dev/sdc* o rozmiarze nie mniejszym niż wskazany rozmiar dziennika.

```

1 [root@messy ~]# mkfs -t xfs -f -l size=10m,logdev=/dev/sdc1 /dev/sda7
2 meta-data=/dev/sda7          isize=256    agcount=4, agsize=33132 blks
3      =                      sectsz=512    attr=2
4 data      =                  bsize=4096   blocks=132528, imaxpct=25
5      =                      sunit=0       swidth=0 blks
6 naming    =version 2        bsize=4096   ascii-ci=0
7 log       =/dev/sdc1        bsize=4096   blocks=2560, version=2
8      =                      sectsz=512    sunit=0 blks, lazy-count=0
9 realtime =none              extsz=4096   blocks=0, rtextents=0

```

Interesujące nas informacje pojawiły się w linii 7. Dziennik został umieszczony na urządzeniu */dev/sdc1*. Jego rozmiar to 2560 bloków po 4096 B, co daje ok 10 MB.

Istotny wpływ na wydajność systemu plików ma rozmiar bloku danych. Im jest on większy, tym mniej operacji należy wykonać w celu przeczytania takiej samej ilości danych. Wydajność systemu wzrasta. Jednak ilość niewykorzystanego miejsca z powodu niecałkowitego wypełnienia bloków danych przechowujących ostatni fragment danych plików wzrasta. Ustalenie innego od domyślnego rozmiaru bloku danych wymaga uruchomienie komendy *mkfs* z opcją **-b** i specyfikatorem *log=* (wartość logarytmu przy podstawie 2) lub *size=* (rozmiar podaje się bezpośrednio w bajtach).

Jednak dla spełnienia ostrych wymagań nałożonych na czas dostępu do informacji przechowywanej w plikach konieczne może stać się założenie sekcji czasu rzeczywistego. Program *mkfs.xfs* należy wówczas uruchomić z opcją **-r** specyfikując po niej wartości atrybutów, do których należą: *rtdev* określający nazwę urządzenia, na którym sekcja zostanie założona, *extsize* definiujący rozmiar bloku danych w sekcji oraz *size* ustalający jej rozmiar, ale jedynie w przypadku, gdy chcemy aby sekcja nie zajmowała całego wskazanego urządzenia.

Na zakończenie utworzymy system plików o wartościach atrybutów jak poprzednio, jednak z blokiem danych o rozmiarze 1024 B oraz sekcją czasu rzeczywistego o rozmiarze 500 MB, założoną na partycji */dev/sdc2* (uprzednio utworzonej).

```

1 [root@messy ~]# mkfs -t xfs -f -l size=10m,logdev=/dev/sdc1 \
2 > -b size=1024 -r rtdev=/dev/sdc2,extsize=4096 /dev/sda7
3 meta-data=/dev/sda7          isize=256    agcount=4, agsize=132529 blks
4      =                      sectsz=512    attr=2
5 data      =                  bsize=1024   blocks=530113, imaxpct=25
6      =                      sunit=0       swidth=0 blks
7 naming    =version 2        bsize=4096   ascii-ci=0
8 log       =/dev/sdc1        bsize=1024   blocks=10240, version=2
9      =                      sectsz=512    sunit=0 blks, lazy-count=0
10 realtime =/dev/sdc2        extsz=4096   blocks=1004062, rtextents=251015

```

Informacja o rozmiarze bloku danych 1024 B, obowiązującym w systemie plików, znajduje się w linii 5. Sekcja czasu rzeczywistego została założona na urządzeniu */dev/sdc2* (linia 10).

Rozmiar wykorzystywanego w niej bloku danych to 4096 B, ich liczba to 251015 co daje ok 1GB (1004062 bloków o rozmiarze 1 kB).

Z umieszczaniem dziennika systemu plików oraz sekcji czasu rzeczywistego poza systemem plików należy być ostrożnym, gdyż nie wszystkie jądra są w stanie obsługiwać taką konfigurację.

### Montowanie systemu plików XFS

Podczas montowania systemu plików XFS, automatycznie odtwarzany jest dziennik tranzakcji. Stanowi to gwarancję spójności systemu plików.

Komenda *mount* posiada wiele opcji charakterystycznych dla systemu plików XFS, które np. umożliwiają określenie położenia dziennika, zdefiniowanie wartości atrybutów przeplotu dysków, podniesienie wydajności czy uruchomienie funkcji specjalnych. Do najczęściej wykorzystywanych opcji należą:

- *logdev* – pozwala wyspecyfikować ścieżkę dostępu do urządzenia, na którym znajduje się zewnętrzny dziennik systemu plików.
- *logbsize* – opcja umożliwia ustawienie rozmiaru buforów dziennika przechowywanych w pamięci operacyjnej. Starsze wersje dopuszczały dwa rozmiary: 16 oraz 32 kB. Obecnie możliwe jest dodatkowo korzystanie z buforów o rozmiarach: 64 kB, 128 kB oraz 256 kB. Wartość domyślna zależy od rozmiaru pamięci RAM systemu komputerowego.
- *logbufs* – oprócz rozmiaru buforów dziennika istnieje możliwość zdefiniowania ich liczby. Zmienna przyjmuje wartości z zakresu 2–8 włącznie.
- *rtdev* – wartością opcji jest ścieżka dostępu do urządzenia, na którym znajduje się sekcja czasu rzeczywistego.
- *dmapi* – opcja włącza wywołania Data Management API lub Data Migration API (DMAPI) umożliwiając przechowywanie systemu plików na urządzeniach pamięci wielopoziomowej.
- *noatime* – użycie opcji spowoduje zablokowanie uaktualniania czasu dostępu do plików, co podnosi wydajność systemu plików.
- *uquota* / *usrquota* / *uqnoenforce* / *quota* – opcja umożliwia rejestrowanie ilości zasobów systemu plików wykorzystywanych przez użytkowników oraz ewentualnie ich ograniczanie.
- *gquota* / *grpquota* / *gqnoenforce* – użycie opcji powoduje rejestrowanie ilości zasobów systemu plików wykorzystywanych przez grupy użytkowników oraz ewentualnie ich ograniczanie.
- *sunit*, *swidth* – umożliwia określenie odpowiednio rozmiaru jednostki przeplotu oraz szerokości paska w przypadku założenia systemu plików na macierzy dyskowej z mechanizmem paskowania (RAID 0). Wartości podaje się jako liczbę bloków o rozmiarze 512 B.
- *nouuid* – opcja powoduje, że podczas podwójnego montowania nie jest sprawdzany identyfikator systemu plików. Opcja jest wykorzystywana podczas montowania migawkowych wolumenów LVM (wykonywanie kopii migawkowych).
- *norecovery* – użycie opcji wymusi zamontowanie systemu plików bez odtwarzania zawartości dziennika. Użycie tej opcji wymusza zamontowanie w trybie tylko do odczytu. Jeśli montowany w ten sposób został niepoprawnie odmontowany, to może się okazać, że niektóre pliki i katalogi są niedostępne.

- *inode64* – w systemie plików XFS zamontowanym z opcją *inode64* i–węzły będą tworzone w dowolnym jego miejscu, nawet jeśli zapisanie jego adresu będzie wymagało użycia więcej niż 32-bitów. Rodzi to pewne problemy podczas tworzenia kopii zapasowej takiego systemu plików.
- *grpuid* / *bsdggroups* oraz *nogrpuid* / *sysvgroups* – opcje umożliwiają określenie sposobu definiowania właściciela grupowego dla nowotworzonego pliku lub katalogu. Opcje *grpuid* lub *bsdggroups* powodują, że jest odziedziczony po katalogu, w którym plik ten lub katalog został utworzony. Opcje *nogrpuid* lub *sysvgroups* (domyślne) spowodują, że właścicielem grupowym zostanie grupa będąca właścicielem procesu, który utworzył plik lub katalog (najczęściej jest to grupa podstawowa właściciela indywidualnego).

Przykładowo, zamontowanie utworzonego na urządzeniu */dev/sda7* systemu plików XFS posiadającego dziennik oraz sekcję czasu rzeczywistego odpowiednio na urządzeniach */dev/sdc1* oraz */dev/sdc2*, w katalogu */scratch1* wymaga użycia następującej komndy:

```
1 [root@messy ~]# mount -t xfs -o logdev=/dev/sdc1,rtdev=/dev/sdc2 /dev/sda7 /scratch1
```

### Kontrola spójności systemu plików

Oprogramowanie narzędziowe systemu plików XFS, dostępne w pakiecie *xfsprogs*, zawiera dwa programy pozwalające na weryfikację systemu plików XFS. Są to: *xfs\_check* wykrywający nieprawidłowości w metadanych systemu plików oraz *xfs\_repair* służący do ich korygowania.

Program *xfs\_check* to skrypt, czyli program dla interpretera poleceń wykorzystujący program *xfs\_db* do analizy stanu systemu plików. Sam proces detekcji nieprawidłowości przebiega bardzo szybko i nie wymaga dodatkowych obszarów pamięci operacyjnej wykorzystywanych na struktury danych w procesie korygowania zawartości systemu plików. Przed użyciem programu *xfs\_check* system plików należy odmontować, gdyż aktywny dziennik oraz operacje zapisu w systemie plików mogą chwilowo stwarzać pozory niespójności.

Program *xfs\_repair* pojawił się nieco później, gdyż duża niezawodność systemu plików XFS nie wymagała jego istnienia. W obecnej implementacji program *xfs\_repair* podejmuje próbę naprawy uszkodzonego systemu plików w siedmiu fazach: Są to:

1. Odnalezienie i sprawdzenie poprawności głównego bloku identyfikacyjnego systemu. Jeśli jego odnalezienie lub weryfikacja z jakiś powodów nie są możliwe, poszukiwane są bloki rezerwowe. Znalezienie pierwszego poprawnego bloku identyfikacyjnego powoduje przepisanie jego zawartości do bloku głównego.
2. Kolejną fazą jest sprawdzenie spójności dziennika i wyzerowanie jego zawartości. Dane przechowywane w dzienniku są wykorzystywane do zweryfikowania poprawności danych zapisanych w strukturach systemu plików.
3. Analiza zawartości każdej z grup alokacji. Weryfikowana jest poprawność listy wolnych i–węzłów i w razie potrzeby uzupełniana o i–węzły nie opisujące żadnego pliku.
4. W każdej z grup alokacji weryfikowana jest poprawność listy wolnych bloków danych oraz rozstrzygane są kwestie zawłaszczenia bloku danych przez kilka plików.
5. W kolejnej fazie, w obrębie każdej z grup alokacji weryfikacji zostają poddane wszystkie, pozostałe struktury danych, jak np. listy zaalokowanych i–węzłów i bloków danych.

6. W całym systemie plików weryfikuje się poprawność struktury katalogów pod kątem przy-  
należenia każdego pliku do konkretnego katalogu. Pliki oraz katalogi nie przypisane do  
żadnego katalogu, łądzą w katalogu *lost+found*. Rozwiązywane są kwestie znajdowania się  
pliku lub katalogu w kilku katalogach.
7. Wprowadzenie korekt z poprzedniej fazy wymusza sprawdzenie i uaktualnienie liczników  
dowiązań do wszystkich plików znajdujących się w systemie plików.

Program *xfs\_repair* posiada kilka opcji, spośród których do najczęściej używanych należą:

- **-l nazwa\_urządzenia\_przechowującego\_dziennik** – opcja wykorzystywana jest do wskaza-  
nia urządzenia przechowującego dziennik naprawianego systemu plików, jeśli znajduje się  
on poza systemem plików.
- **-n** – użycie opcji powoduje, że program będzie raportował wykryte błędy ale nie podejmie  
żadnych kroków naprawczych. Istotna, jeśli kolejność naprawiania błędów ma wpływ na  
stan systemu plików.

Dla zbadania jakości obu programów uszkodzono utworzony na urządzeniu */dev/sda7* system  
plików, zapisując na jego boczku jeden blok o wartości losowej i rozmiarze 8192 B. Użyta  
komenda *dd* miała następującą postać:

```
1 dd if=/dev/urandom of=/dev/sda7 bs=8192 count=1
```

Uruchomienie programu *xfs\_check* przyniosło następujący wynik:

```
1 [root@messy fsck]# xfs_check /dev/sda7
2 xfs_check: /dev/sda7 is not a valid XFS filesystem (unexpected SB magic number 0x6b4fd364)
3 /usr/bin/xfs_check: line 28: 3209 Floating point exceptionxfs_db$DB0PTS -F -i -p xfs_check -
```

Próba jego zamontowania zakończyła się niepowodzeniem:

```
1 [root@messy fsck]# mount -t xfs /dev/sda7 /scratch1
2 mount: wrong fs type, bad option, bad superblock on /dev/sda7,
3     missing codepage or helper program, or other error
4     In some cases useful info is found in syslog - try
5     dmesg | tail  or so
```

Program *xfs\_repair* dokonał próby naprawy systemu plików:

```
1 [root@messy fsck]# xfs_repair /dev/sda7
2 Phase 1 - find and verify superblock...
3 bad primary superblock - bad magic number !!!
4
5 attempting to find secondary superblock...
6 .....
7 verified secondary superblock...
8 writing modified primary superblock
9 sb realtime bitmap inode 18446744073709551615 (NULLFSINO) inconsistent with calculated value
10 resetting superblock realtime bitmap ino pointer to 65
11 sb realtime summary inode 18446744073709551615 (NULLFSINO) inconsistent with calculated value
```

### 3.4. ADMINISTRACJA WYBRANYMI SYSTEMAMI PLIKÓW W DYSTRYBUCJI REDHAT231

```
12 resetting superblock realtime summary ino pointer to 66
13 Phase 2 - using internal log
14     - zero log...
15     - scan filesystem freespace and inode maps...
16 bad magic # 0x8e04e1b for agf 0
17 bad version # 1154020038 for agf 0
18 bad sequence # -832137946 for agf 0
19 bad length 1986107658 for agf 0, should be 132529
20 flfirst -640728869 in agf 0 too large (max = 128)
21 fllast 826267205 in agf 0 too large (max = 128)
22 bad magic # 0xe76447e3 for agi 0
23 bad version # 1711549882 for agi 0
24 bad sequence # -1637797034 for agi 0
25 bad length # -1184629977 for agi 0, should be 132529
26 reset bad agf for ag 0
27 reset bad agi for ag 0
28 bad agbno 3496059205 in agfl, agno 0
29 freeblk count 1 != flcount -1462755768 in ag 0
30 bad agbno 2103462874 for btbno root, agno 0
31 bad agbno 924778508 for btbcnt root, agno 0
32 bad agbno 1582374067 for inobt root, agno 0
33 root inode chunk not found
34 Phase 3 - for each AG...
35     - scan and clear agi unlinked lists...
36 error following ag 0 unlinked list
37     - process known inodes and perform inode discovery...
38     - agno = 0
39     - agno = 1
40     - agno = 2
41     - agno = 3
42     - process newly discovered inodes...
43 Phase 4 - check for duplicate blocks...
44     - setting up duplicate extent list...
45     - check for inodes claiming duplicate blocks...
46     - agno = 0
47     - agno = 1
48     - agno = 2
49     - agno = 3
50 Phase 5 - rebuild AG headers and trees...
51     - reset superblock...
52 Phase 6 - check inode connectivity...
53     - resetting contents of realtime bitmap and summary inodes
54     - traversing filesystem ...
55     - traversal finished ...
56     - moving disconnected inodes to lost+found ...
57 Phase 7 - verify and correct link counts...
58 Note - stripe unit (0) and width (0) fields have been reset.
59 Please set with mount -o sunit=<value>,swidth=<value>
```

„Poprawiony” system plików można było zamontować i odzyskać zapisane w nim dane.

### Wybrane programy narzędziowe

Znalezienie systemu Linux zawierającego w dystrybucji programy narzędziowe dla systemu plików XFS jest kłopotliwe. Stąd najczęściej oprogramowanie to ściąga się w postaci kodu źródłowego, a następnie kompiluje w systemie docelowym. Najistotniejszymi pakietami są: *xfs\_progs* zawierający podstawowe programy narzędziowe do zakładania systemu plików, zmiany rozmiaru, sprawdzania jego integralności oraz podejmowania prób naprawy, *acl* zawierający obsługę list kontroli dostępu, *attr* dostarczający narzędzi do obsługi rozszerzonych atrybutów plików, *xfs-dump* zawierający programy narzędziowe do tworzenia kopii systemu plików oraz odtwarzania jego stanu z kopii zapasowej.

Pakiet *xfs\_progs* zawiera następujące programy użytkowe:

- *fsck.xfs* – jak podaje strona podręcznika systemowego program ten nie robi nic. Jego wykonanie zawsze kończy się sukcesem. W pakiecie znalazł się jedynie aby zapewnić zgodność z ogólnymi konwencjami nazewniczymi dla programów badających spójność systemu plików.
- *mkfs.xfs* – program służy do zakładania systemu plików XFS. Domyślnie znajduje się on w katalogu */sbin* i jest uruchamiany przez program *mkfs* uruchomiony z opcją *-t xfs*.
- *xfs\_admin* – to program dla interpretera poleceń (skrypt), który korzysta z programu *xfs\_db* do analizy oraz modyfikacji wartości atrybutów systemu plików. System plików powinien zostać uprzednio odmontowany lub zamontowany w trybie tylko do odczytu.
- *xfs\_bmap* – program tworzy i wypisuje listę bloków danych zajmowanych przez dane plików znajdujących się w systemie.
- *xfs\_check* – sprawdza spójność systemu plików.
- *xfs\_db* – służy do analizy i modyfikacji systemu plików XFS. Raczej nie jest on bezpośrednio uruchamiany. Jego wywołanie znajduje się w licznych skryptach.
- *xfs\_freeze* – program służy do blokowania lub odblokowywania dostępu do systemu plików. Przykładowo jest on uruchamiany przed wykonaniem kopii zapasowej, gdyż opóźnia operacje zapisu zapewniając chwilowo spójny stan system plików.
- *xfs\_growfs* – umożliwia zmianę rozmiaru istniejącego systemu plików.
- *xfs\_info* – umożliwia uzyskanie informacji o zamontowanym systemie plików. Służy także weryfikacji parametrów wykorzystywanych przez program *xfs\_growfs*.
- *xfs\_logprint* – program wypisuje zawartość dziennika systemu plików. Wykorzystywany podczas analizy błędów i niespójności systemu plików.
- *xfs\_ncheck* – umożliwia uzyskanie listy wykorzystanych i-węzłów oraz ścieżek dostępu do plików opisywanych przez te i-węzły.
- *xfs\_repair* – program do usuwania niespójności oraz uszkodzeń systemu plików.
- *xfs\_rtcp* – program do kopiowania plików do sekcji czasu rzeczywistego systemu plików XFS.



Podstawowe programy znajdujące się w pakiecie *acl* to:

- *getfacl* – wypisuje listę kontroli dostępu wskazanych plików i katalogów.
- *setfacl* – program służy do tworzenia listy kontroli dostępu dla plików i katalogów.
- *chacl* – umożliwia tworzenie oraz modyfikację istniejących list kontroli dostępu plików i katalogów.

W pakiecie *attr* znajduje się program *attr*, który umożliwia tworzenie, dodawanie oraz usuwanie rozszerzonych atrybutów plików.

- *xfs\_copy* – program służy do kopiowania całego systemu plików na inne urządzenie (jedno lub kilka). Ze względu na fakt, iż operacja ta może być rozciągnięta w czasie, dla zagwarantowania niezmienności systemu plików wymagane jest jego odmontowanie.
- *xfsdq* – to narzędzie pomocnicze do reglamentowania przydzielania zasobów systemu plików użytkownikom. Umożliwia uzyskanie zrzutu informacji o kontyngentach w obrębie danego systemu plików.
- *xfsrq* – umożliwia ustawienie kontyngentów dyskowych w oparciu o plik uzyskany dzięki programowi *xfsdq*.
- *xfsdump* – program umożliwia tworzenie kopii systemu plików. Jest on rozszerzeniem standardowego, linuksowego programu *dump* uwzględniającym specyficzne dla systemu plików XFS dane takie, jak listy kontroli dostępu czy rozszerzone atrybuty plików.
- *xfs\_estimate* – program pozwala oszacować ile miejsca zająłby istniejący katalog w systemie plików XFS. Jest to wykorzystywane podczas przenoszenia danych do systemu plików XFS.
- *xfs\_fsr* – reorganizator systemu plików. Program działa na zamontowanym systemie plików starając się zapewnić optymalne ze względu na czas dostępu rozmieszczenie plików. Jest to zatem zwykły defragmentator. Czas jego działania pozostawia wiele do życzenia.
- *xfsinutil* – program działa na bazie danych kopii przyrostowych tworzonej przez program *xfs\_dump*. Weryfikuje jej spójność usuwając w razie potrzeby zbędne wpisy.
- *xfsrestore* – program umożliwia odtworzenie systemu plików z kopii zapasowej stworzonej programem *xfsdump*. Umożliwia odtworzenie z pojedynczej kopii, jak również z kopii tworzonych przyrostowo.

### 3.4.6 Journaled File System (JFS)

Journaled File System (JFS) został opracowany przez firmę IBM z początkowym przeznaczeniem dla systemów operacyjnych AIX. Następnie pojawiła się jego implementacja dla systemu operacyjnego OS/2. Jego katastrofa komercyjna spowodowała ponowne przepisanie kodu systemu plików dla systemu operacyjnego AIX. Kod ten stał się podstawowym dla systemu plików JFS, opracowywanego dla systemów operacyjnych z rodziny Linux. Długotrwała eksploatacja systemu plików w systemach komputerowych o różnym przeznaczeniu pozwoliła na wyeliminowanie błędów i uczyniła go jednym z bardziej niezawodnych. Strona projektu znajduje się pod adresem <http://jfs.sourceforge.net/>.

JFS jest 64 bitowym systemem plików dającym możliwość uzyskania przestrzeni adresowej o rozmiarze od 512 TB do 32 PB, w zależności od wybranego podczas jego zakładania rozmiaru

bloku danych. Możliwe do zastosowania rozmiary bloków danych to 512, 1024, 2048 oraz 4096 B. Jednak w implementacji dla systemów linuksowych dostępne są jedynie bloki danych o rozmiarze 4096 B. Dalsze ograniczenia, wynikające przykładowo z maksymalnego rozmiaru urządzenia blokowego obsługiwane przez jądro powodują, iż maksymalny, rzeczywisty rozmiar systemu plików JFS wynosi 4 PB (dla jądra 2.6). System ten posiada także swój rozmiar minimalny, wynoszący 16 MB. Przy pojemnościach obecnie produkowanych pamięci nie stanowi to żadnego problemu, nawet w przypadku implementacji specjalizowanej minidystrybucji.

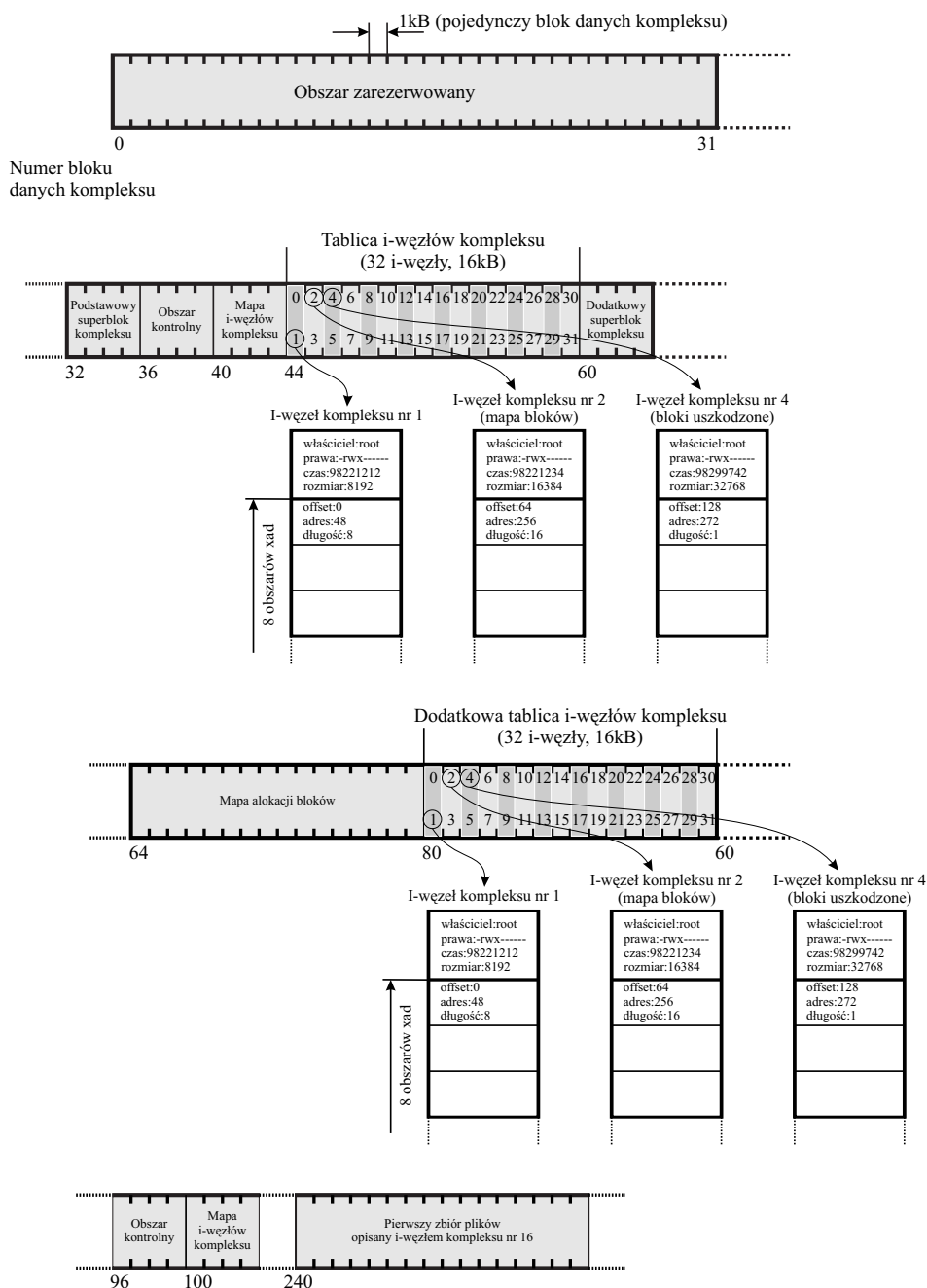
### Budowa wewnętrzna

W dostępnych materiałach opisujących budowę systemu plików JFS terminologia może zaskoczyć osobę, która wcześniej nie miała do czynienia z dokumentacją firmy IBM. Stąd niniejszy podpunkt, oprócz opisu budowy wewnętrznej zawierał będzie definicje pojęć.

System plików JFS definiuje pojęcia dostępnej przestrzeni dyskowej jako zespół lub kompleks (*aggregate*) oraz zbioru czy zestawu plików (*filesystems*) zawierającego drzewo katalogów, którego zawartość zostaje udostępniona przez jego zamontowanie. W pierwszych wersjach JFS dopuszczał zorganizowanie jednego kompleksu na jednej partycji dyskowej oraz jednego zestawu plików w obrębie jednego kompleksu. Obecnie istnieje możliwość stworzenia wielu zestawów w jednym zespole. Znaczenie pojęć kompleks oraz zestaw plików jest takie samo jak w rozproszonych systemach plików, przykładowo w Distributed Computing Environment Distributed Filesystems (DCE DFS). Wynika to z faktu, iż na etapie projektowania przewidziano możliwość rozszerzania struktur metadanych.

Zarówno zespoły jak i zestawy plików posiadają własne superbloki oraz inne struktury opisujące organizację przestrzeni dyskowej. Rozmiar bloku danych dla ich przechowywania jest w obrębie kompleksu stały i może wynosić 512, 1024, 2048 lub 4096 B. Definiuje on rozmiar najmniejszej jednostki alokacji zarządzanej przez kompleks. Nie jest nigdy mniejszy od rozmiaru bloku danych partycji na której kompleks został założony. Struktury kompleksu przedstawiono na rysunku 3.15. Jak widać tworzą go:

- Blok o rozmiarze 32 kB, który rozpoczyna kompleks. Jest to obszar zarezerwowany, wykorzystywany przez niektóre programy narzędziowe.
- Kolejną strukturą jest podstawowy blok identyfikacyjny kompleksu (superblok kompleksu) (*Primary Aggregate Superblock*). Zawiera on informacje o wartościach atrybutów kompleksu, takich jak jego rozmiar, rozmiar grup alokacji czy rozmiar bloku danych. Dodatkowy blok identyfikacyjny (*Secondary Aggregate Superblock*) stanowi dokładną kopię pierwszego i jest wykorzystywany jeśli blok podstawowy ulegnie uszkodzeniu. Oba bloki znajdują się zawsze w stałym miejscu, dzięki czemu nie ma problemów z ich odnalezieniem.
- Tablica i-węzłów kompleksu (*Aggregate Inode Table*) zawiera i-węzły opisujące struktury danych i metadanych kompleksu. Strukturą danych, w której zapisane są i-węzły jest tablica. Kompleks nie ma budowy hierarchicznej, jak np. drzewo katalogów.
- Dodatkowa tablica i-węzłów kompleksu (*Aggregate Inode Table*) zawiera kopię i-węzłów z tablicy kompleksu. Duplikowanie informacji wynika z faktu, iż i-węzły kompleksu zawierają informacje konieczne do odszukiwania struktur opisujących znajdujące się na nim systemy plików. Duplikowaniu podlegają jedynie adresy umożliwiające odnalezienie struktur, a nie same struktury.
- Mapa i-węzłów kompleksu (*Aggregate Inode Map*) opisuje tablicę i-węzłów kompleksu. Zawiera ona informację o tym, które i-węzły kompleksu są wolne, co przyspiesza działanie funkcji wyszukiwania.



Rysunek 3.15: Schemat logiczny systemu plików JFS. Rysunek przedstawia organizację struktur opisujących kompleks, na którym znajdują się zestawy plików.

- Dodatkowa mapa i-węzłów kompleksu (*Secondary Aggregate Inode Map*) opisuje dodatkową tablicę i-węzłów kompleksu. Ponieważ stanowi ona kopię tablicy i-węzłów kompleksu, więc dodatkowa mapa i-węzłów może być wykorzystywana do opisu obu tablic.

- Mapa alokacji bloku (*Block Allocation Map*) jest strukturą, która zawiera listy wolnych oraz zajętych bloków danych kompleksu.
- Obszar danych programu *fsck* (nie przedstawiona na rysunku 3.15). Jego istnienie jest naturalną konsekwencją istnienia w systemie plików JFS kompleksów o bardzo dużych rozmiarach. Może się okazać, że podczas weryfikowania spójności kompleksu program *fsck* nie będzie w stanie przechować w pamięci informacji o nim i konieczne stanie się jej zapisanie w systemie plików. W tym celu zarezerwowano ten właśnie obszar. Jest on dostępny przez wskaźnik znajdujący się w bloku identyfikacyjnym kompleksu i znajduje się zawsze na końcu kompleksu.
- Dziennik kompleksu (*In-line Log*) (nie przedstawiony na rysunku 3.15) służy dziennikowaniu operacji przeprowadzanych na meta danych kompleksu. Obszar ten jest dostępny przez wskaźnik znajdujący się w bloku identyfikacyjnym partycji.

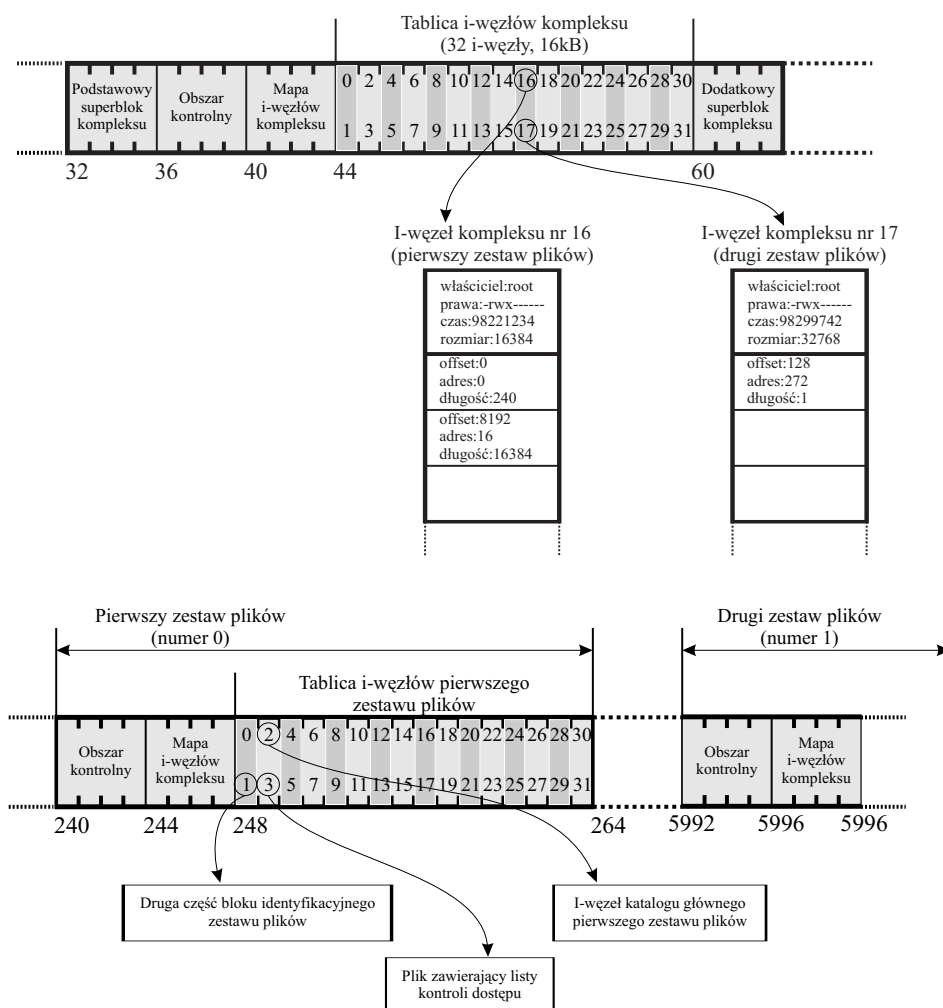
Podczas zakładania kompleksu tworzony jest jedynie jego pierwszy i-węzeł. Kolejne pojawiają się dynamicznie. Opisują one pewne struktury kompleksu. I tak:

- I-węzeł o numerze 0 kompleksu jest zarezerwowany. Może być wykorzystywany do przechowywania tymczasowej informacji przez programy narzędziowe.
- Pierwszy i-węzeł kompleksu opisuje blok dyskowy kompleksu zawierający mapę i-węzłów kompleksu. Jego położenie 4 kB za podstawowym blokiem identyfikacyjnym kompleksu powoduje, że jest on łatwy do odnalezienia, a wraz z nim tablica i-węzłów kompleksu.
- Dla utworzenia kopii tablicy i-węzłów kompleksu system plików JFS potrzebuje odnaleźć kopię pierwszego i-węzła kompleksu dla odnalezienia pozostałych części kopiowanych tablic. Blok opisu kompleksu zawiera deskryptor będący adresem pierwszego i-węzła w dodatkowej tablicy i-węzłów kompleksu. Stąd system plików JFS uzyska dostęp do drugiego i-węzła kompleksu oraz pozostałej części dodatkowej tablicy i-węzłów kompleksu.
- Drugi i-węzeł kompleksu opisuje mapę alokacji bloków danych.
- Trzeci i-węzeł kompleksu opisuje dziennik kompleksu.
- Czwarty i-węzeł kompleksu opisuje bloki, które podczas zakładania systemu plików zostały zaznaczone jako uszkodzone. Budują one plik regularny, dzięki czemu w liście bloków figurują jako zajęte i nie zostaną wykorzystane do budowy innych plików.
- I-węzły o numerach od 5 do 15 są zarezerwowane dla przyszłych rozszerzeń, w celu zapewnienia wstecznej zgodności systemów plików.
- I-węzły od numeru 16 reprezentują zestawy plików. Są to tzw. i-węzły alokacji zestawu plików (*Fileset Allocation Map Inode*), których zawartość umożliwia dostęp do danych opisujących zestaw plików. I-węzły te są tworzone dynamicznie w momencie tworzenia nowego zestawu plików w danym kompleksie.

Przestrzeń dyskowa zajmowana przez kompleks jest podzielona na jednostki zwane grupami alokacji (*allocation groups*, AG). Rozwiązanie to pozwala na stosowanie zaawansowanych metod alokacji, co przekłada się bezpośrednio na wydajność systemu plików. Stosowana jest przede wszystkim koncepcja lokalności, która wymusza aby dane i opisujące je i-węzły znajdowały się na dysku w bliskim sąsiedztwie. Z drugiej strony dane od siebie niezależne umieszczane są w kompleksie w zależności od dostępnego miejsca. Istotny jest także rozmiar grupy alokacji, który

ma wpływ na defragmentację przestrzeni. Maksymalna liczba grup alokacji została ograniczona w obrębie kompleksu do 128. Minimalna liczba bloków danych kompleksu tworzących grupę alokacji wynosi 8192. Wartości te są przechowywane w bloku identyfikacyjnym kompleksu.

Zestaw plików jest zbiorem plików i katalogów będących niezależnym, możliwym do zamontowania fragmentem drzewa katalogów. Jak wspomniano, w brębie kompleksu może istnieć wiele zestawów plików wykorzystujących do opisu przechowywanych danych jego struktury. Nie jest natomiast możliwe, aby jeden zestaw plików należał do dwóch lub więcej kompleksów. Struktury opisujące zestaw plików przedstawiono na rysunku 3.16.



Rysunek 3.16: Schemat logiczny systemu plików JFS. Rysunek przedstawia organizację struktur opisujących zestaw plików.

Zestaw plików opisuje tablica i-węzłów zestawu plików (*Filesset Inode Table*), zawierająca definicje atrybutów struktur organizujących zestaw. Strukturą pomocniczą jest mapa alokacji i-węzłów zestawu (*Filesset Inode Allocation Map*). Zawiera ona informacje o wykorzystaniu i-węzłów zestawu, co przyspiesza operacje tworzenia nowych struktur w obrębie zestawu. Dla podniesienia poziomu bezpieczeństwa tablica i-węzłów zestawu posiada kopię. Tzw. super i-węzeł (i-węzeł

główny) opisuje mapę alokacji oraz zawiera wartości podstawowych atrybutów zestawu (jego rozmiar, wykorzystane zasoby, itd.). Zakładanie zestawu plików powoduje utworzenie obszaru pierwszego i-węzła. Pozostałe będą tworzone dynamicznie, w trakcie pracy systemu plików. Kolejne i-węzły zestawu plików zawierają:

- I-węzeł o numerze 0 jest zarezerwowany.
- Dodatkowe informacje, które nie znalazły się w tablicy i-węzłów kompleksu oraz mapie alokacji i-węzłów zestawu zapisane zostały w i-węźle numer 1.
- Katalog główny zestawu plików opisuje i-węzeł o numerze 2. To dla zachowania konwencji przyjętej w systemach uniksowych, a mówiącej, że katalog główny opisuje i-węzeł numer 2.
- I-węzeł numer 3 opisuje plik zawierający listy kontroli dostępu obiektów znajdujących się w zestawie plików.
- I-węzły od numeru 4 opisują pliki użytkowników, katalogi, pliki będące dowiązaniami symbolicznymi oraz inne obiekty zestawu plików.

Plik w systemie plików JFS jest przechowywany w obszarach (*extent*). Obszar to sekwencja znajdujących się obok siebie bloków danych, których liczba jest różna i zależy od aktualnej defragmentacji przestrzeni dyskowej. Wykorzystane bloki mogą należeć do kilku grup alokacji. Koncepcja ta jest analogiczna do zastosowanej w systemie plików XFS (patrz rysunek 3.14), z dokładnością do ich definiowania polegającej na podaniu numeru pierwszego, budującego go bloku oraz ich liczby. Liczba bloków budujących obszar jest zapisana na 24 bitach. Stąd jej maksymalna wartość wynosi  $2^{24} - 1$ . Przy największym, obsługiwanym przez system plików JFS rozmiarze bloku danych wynoszącym 4096 B umożliwia to przechowywanie pliku o rozmiarze do 64 GB. Każdy obszar jest indeksowany z wykorzystaniem B+ drzewa, co znacznie przyspiesza operacje wyszukiwania oraz wstawiania nowego obszaru między aktualnie wykorzystywane. Zwiększenie efektywności dostępu do danych uzyskuje się także dzięki procedurom wyszukującym wolne obszary. Ogólnie przyjmuje się, że ich liczba dla pojedynczego pliku powinna być minimalna.

Rozmiar i-węzła w systemie plików JFS wynosi 512 B. I-węzeł przechowywany na dysku zawiera cztery podstawowe grupy informacji. Pierwsza opisuje wymagane odpowiednią normą POSIX atrybuty obiektu przechowywanego w systemie. Druga grupa zawiera dodatkowe atrybuty, między innymi zapewniające przenaszalność systemu plików między różnymi systemami operacyjnymi oraz nagłówki dla B+ drzewa. W grupie trzeciej znajduje się informacja o położeniu obszaru zawierającego korzeń B+ drzewa oraz wartości podstawowych atrybutów obiektu. Grupa czwarta zawiera definicję pozostałych atrybutów obiektu. Jak wspomniano, i-węzły alokowane są dynamicznie, w miarę potrzeb, a nie jak w przypadku systemów plików z rodziny ext przy ich zakładaniu. I-węzły umieszczane są we wpisach stanowiących ciągły obszar dysku. Obszar może zawierać maksymalnie 32 i-węzły. Stąd jego obszar wynosi 16 kB. Zalety takiego rozwiązania są następujące:

- I-węzły mogą zostać utworzone w dowolnym miejscu systemu plików. Numer i-węzła nie wynika z jego położenia, co upraszcza organizację kompleksów oraz zestawów plików. W trakcie pracy systemu plików i-węzły mogą być przenoszone do innych jego obszarów z zachowaniem numeru. Stąd zmiana organizacji danych na dysku nie wymaga uaktualniania numerów i-węzłów. Brak powiązania między numerem i-węzła, a jego położeniem w systemie plików jest konieczne dla umożliwienia klonowania rozproszonych systemów plików (*Distributed File Systems*). Tylko wówczas kopiowany i-węzeł jesteśmy w stanie umieścić w dowolnym miejscu systemu plików z zachowaniem jego numeru.

- Alokacja dla dużych plików uwzględnia wykorzystanie ciągłych obszarów alokacji. W przypadku utworzenia statycznej listy i-węzłów jest to znacznie trudniejsze do osiągnięcia.
- Umożliwia racjonalne gospodarowanie przestrzenią systemu plików, gdyż jest ona wykorzystywana do tworzenia relatywnie dużych i-węzłów (512 B) tylko wówczas, jeśli jest to konieczne.

Dynamiczne tworzenie i-węzłów posiada także swoje wady. Przede wszystkim wymagana jest dodatkowa struktura opisująca ich położenie na dysku. W rozwiązaniu ze statyczną listą i-węzłów ich położenie jest zdeterminowane. Dodatkowo, struktura ta ma decydujący wpływ na integralność systemu plików. Stąd wymagane są specjalne zabiegi, jak np. tworzenie jej kopii oraz zachowanie spójności kopii, w celu podniesienia poziomu bezpieczeństwa systemu plików, co natychmiast odbija się negatywnie na jego wydajności.

System plików JFS wykorzystuje technologie stosowane w bazach danych do gromadzenia informacji o operacjach na metadanych. Są one traktowane jako operacje atomowe. W przypadku awarii systemu komputerowego, spójność systemu plików jest przywracana w oparciu o informacje zapisane w dzienniku, co jest operacją zajmującą znacznie mniej czasu niż przeglądanie całego systemu plików wykonywane odpowiednim programem *fsck*.

### Zakładanie systemu plików JFS

Programy narzędziowe dla systemu plików JFS są dostępne w pakiecie *jfsutils* w postaci archiwum kodów źródłowych lub skompilowanych programów spakowanych do formatu wykorzystywanego przez program *rpm*. Zalecane jest oczywiście ściągnięcie kodów źródłowych (np. ze strony <http://jfs.sourceforge.net/>) oraz ich kompilacja w systemie, w którym będą one wykorzystywane. W pliku *README* znajduje się dokładna instrukcja postępowania, która jest klasyczną w takich przypadkach.

Okazuje się, że pakiet *jfsutils* zawiera niewielką liczbę programów, z których parę służy do celów innych, niż diagnostyka systemu plików. Do utworzenia systemu plików służy program *mkfs.jfs*, który w wyniku instalacji zostaje skopiowany do katalogu */sbin* i jest uruchamiany przez program *mkfs* z opcją *-t* o wartości *jfs*. Wymagane jest jeszcze podanie ścieżki dostępu do pliku urządzenia, na którym system plików ma zostać założony:

```

1 [root@messy jfsutils-1.1.14]# mkfs -t jfs /dev/sda7
2 mkfs.jfs version 1.1.14, 06-Apr-2009
3 Warning! All data on device /dev/sda7 will be lost!
4
5 Continue? (Y/N) y
6 |
7 Format completed successfully.
8
9 530113 kilobytes total disk space.
```

Jak widać program *mkfs.jfs* nie udziela zbyt wielu informacji o wartościach atrybutów utworzonego systemu plików. Zamontujmy utworzony system plików i sprawdźmy jego dostępne zasoby:

```

1 [root@messy jfsutils-1.1.14]# mount -t jfs /dev/sda7 /scratch1
2 [root@messy jfsutils-1.1.14]# df /dev/sda7
3 Filesystem          1K-blocks      Used Available Use% Mounted on
```

```

4 /dev/sda7          526816      200    526616    1% /scratch1
5 [root@messy jfsutils-1.1.14]# df -i /dev/sda7
6 Filesystem          Inodes    IUsed   IFree IUse% Mounted on
7 /dev/sda7          1053248      4 1053244    1% /scratch1

```

Dla porównania, na tej samej partycji założono system plików ext3. Wykorzystano wartości domyślne programu *mkfs.ext3*. Udostępniane przez utworzony system plików zasoby są następujące:

```

1 [root@messy jfsutils-1.1.14]# mount -t ext3 /dev/sda7 /scratch1
2 [root@messy jfsutils-1.1.14]# df /dev/sda7
3 Filesystem          1K-blocks    Used Available Use% Mounted on
4 /dev/sda7          521748    16816   478428    4% /scratch1
5 [root@messy jfsutils-1.1.14]# df -i /dev/sda7
6 Filesystem          Inodes    IUsed   IFree IUse% Mounted on
7 /dev/sda7          33200      11   33189    1% /scratch1

```

Porównując uzyskane komendą *df* raporty możemy stwierdzić, że rozmiar udostępnianej przestrzeni adresowej różni się znacząco na korzyść systemu plików JFS. Dodatkowo, podstawowy rozmiar samego systemu plików jest również nieco większy w przypadku systemu plików JFS, mimo iż oba systemy plików zostały założone na tej samej partycji. Różnice te mają dwie przyczyny. Przede wszystkim system plików ext3 wykorzystuje część przestrzeni systemu plików do wstępnej alokacji i-węzłów i innych struktur danych, które w systemie plików JFS są alokowane na bieżąco, w miarę potrzeb. Dodatkowo system plików ext3 rezerwuje domyślnie 5% swojej przestrzeni na bloki danych, które mogą być wykorzystywane jedynie przez użytkownika *root* na potrzeby prowadzonych przez niego prac administracyjnych, jeśli dotępna dla pozostałych użytkowników przestrzeń uległa wyczerpaniu. W systemie plików JFS obszar ten nie jest rezerwowany.

U uzupełnieniu, program *mkfs.jfs* posiada kilka opcji umożliwiających określanie wartości jego atrybutów. Do najczęściej używanych należą:

- **-c** – opcja wymusza przeszukanie urządzenia, na którym zakładany jest system plików w celu znalezienia uszkodzonych bloków.
- **-j nazwa\_pliku\_urządzenia** – opcja umożliwia wskazanie urządzenia, na którym zostanie umieszczony plik dziennika.
- **-J opcje\_dziennika** – oprócz wskazania urządzenia, na którym ma znajdować się dziennik, niekiedy konieczne jest nadanie wartości innym jego atrybutom. Przykładem może być rozmiar dziennika. Umożliwia to właśnie ta opcja.
- **-L etykieta** – użycie opcji umożliwia nadanie etykiety zakładanemu systemowi plików. Można ją wykorzystać do wskazania systemu plików dla jego zamontowania.
- **-q** – opcja powoduje brak konieczności potwierdzenia operacji zakładania systemu plików.
- **-s rozmiar\_pliku\_dziennika** – opcja umożliwia ustalenia rozmiaru pliku dziennika. Jednostką są MB. Wartość domyślna to 0.4% rozmiaru kompleksu.



### Określanie rozmiaru i położenia dziennika

Dziennik systemu plików JFS jest wspólny dla wszystkich zestawów plików znajdujących się w danym kompleksie. Jego domyślny rozmiar wynosi 0.4% przestrzeni kompleksu. Jednak w przypadku systemu plików, w którym aplikacje tworzą i usuwają duże ilości plików tymczasowych lub jest on eksploatowany przez dużą liczbę użytkowników jednocześnie, konieczne staje się zwiększenie rozmiaru dziennika. Dla przykładu założymy system plików na urządzeniu opisanym plikiem `/dev/sda7` posiadający dziennik zewnętrzny na urządzeniu `/dev/sdc1` o rozmiarze ok. 40 MB. W takim przypadku wymagane jest w pierwszym kroku utworzenie partycji o odpowiednim rozmiarze, a następnie założenie na niej dziennika:

```
1 [root@messy jfsutils-1.1.14]# jfs_mkfs -J journal_dev /dev/sdc1
2 jfs_mkfs version 1.1.14, 06-Apr-2009
3 |
4 JFS external journal created SUCCESSFULLY on /dev/sdc1.
```

W kolejnym kroku tworzymy system plików, wskazując położenie dziennika:

```
1 [root@messy jfsutils-1.1.14]# mkfs -t jfs -J device=/dev/sdc1 /dev/sda7
2 mkfs.jfs version 1.1.14, 06-Apr-2009
3 Warning! All data on device /dev/sda7 will be lost!
4
5 Continue? (Y/N) y
6
7 Format completed successfully.
8
9 530113 kilobytes total disk space.
```

Zasoby utworzonego systemu plików są następujące:

```
1 [root@messy jfsutils-1.1.14]# df /dev/sda7
2 Filesystem          1K-blocks      Used Available Use% Mounted on
3 /dev/sda7           529888         200   529688   1% /scratch1
4 [root@messy jfsutils-1.1.14]# df -i /dev/sda7
5 Filesystem          Inodes        IUsed   IFree IUse% Mounted on
6 /dev/sda7          1059392         4 1059388    1% /scratch1
```

Jak należało się spodziewać, umieszczenie dziennika na innym urządzeniu niż system plików spowodowało, iż ilość dostępnych zasobów, zarówno bloków danych jak i i-węzłów wzrosła.

### Montowanie systemu plików JFS

Oprócz opcji ogólnych, możliwych do wykorzystania w przypadku montowania każdego systemu plików, program `mount` dla systemu plików JFS udostępnia kilka opcji dedykowanych. Należą do nich:

- **iocharset=*nazwa*** – opcja umożliwia zdefiniowanie strony kodowej, użytej do konwersji z unikodu. Domyślnie konwersja nie jest wykonywana.

- **resize=value** – użycie opcji umożliwia zmianę dostępnej liczby bloków danych systemu plików do liczby podanej jako wartość opcji. Możliwe jest jedynie zwiększenie liczby dostępnych bloków. Użycie opcji bez podania wartości powoduje zwiększenie rozmiaru systemu plików do całego rozmiaru partycji, na której został on założony.
- **nointegrity** – opcja wyłącza zapisywanie zmian metadanych w dzienniku systemu plików. Podnosi to wydajność systemu plików, ale jest stosowane sporadycznie. Przykładem może być odtwarzanie zawartości systemu plików z kopii zapasowej.
- **integrity** – jest to opcja domyślna. Umożliwia zapisywanie zmian metadanych systemu plików w dzienniku.
- **errors=continue**, **errors=remount-ro** lub **errors=panic** – opcja określa zachowanie systemu plików w przypadku wystąpienia błędów. Kolejne wartości oznaczają: zignorowanie błędu, przemontowanie systemu do trybu tylko do odczytu lub zatrzymanie jego pracy.
- **noquota**, **quota**, **usrquota**, **ngroupquota** – opcje te są ignorowane, gdyż implementacje dla systemów linuksowych nie umożliwiają reglamentowania zasobów systemu plików JFS.

### Kontrola spójności systemu plików

Programem przeznaczonym do badania spójności systemu plików JFS jest znajdujący się w pakiecie *jfsutils* program *fsck.jfs*, który podczas instalacji jest umieszczany w katalogu */sbin*. Dla ścisłości, plik ten jest widziany także pod nazwą *jfs\_fsck* (dowiązanie twarde). Może on zostać uruchomiony bezpośrednio lub przez program *fsck* wywołany z opcją **-t** i wartością *jfs*. W obu przypadkach wymagane jest podanie nazwy pliku urządzenia, na którym znajduje się weryfikowany system plików. Jeśli dziennik systemu plików znajduje się na innym urządzeniu, to należy je wskazać korzystając z opcji **-j**. Zalecane jest także wcześniejsze odmontowanie weryfikowanego systemu plików:

```

1 [root@messy ~]# fsck.jfs -j /dev/sdc1 /dev/sda7
2 fsck.jfs version 1.1.14, 06-Apr-2009
3 processing started: 11/11/2009 15.22.42
4
5 /dev/sda7 is mounted.
6
7 WARNING!!!
8 Running fsck on a mounted file system
9 may cause SEVERE file system damage.
10
11 Do you really want to continue (y/n)? y
12 yes
13
14 Using default parameter: -p
15 The current device is: /dev/sda7
16 Block size in bytes: 4096
17 Filesystem size in blocks: 132528
18 **Phase 0 - Replay Journal Log
19 Filesystem is clean.
```

Uruchomiony w ten sposób program *fsck.jfs* poinformował o użyciu opcji *-p* (linia 14). Jest ona opcją domyślną i powoduje, że zostaną podjęte próby automatycznego naprawiania napotkanych niespójności. W takiej sytuacji, pierwszą czynnością jest sięgnięcie do dziennika systemu plików i w oparciu o jego zawartość uaktualnienie metadanych. Program *fsck.jfs* nie przejdzie do kolejnych faz jeśli odczyt zawartości dziennika będzie niemożliwy lub nie uda się przy jego pomocy przywrócić spójności kompleksu. W powyższym przykładzie został przetestowany system plików, który po utworzeniu został zamontowany i skopiowano do niego ok. 100 MB danych. Jak widać jest on spójny. Uszkodzimy go kopiując na jego początek bajty o wartościach przypadkowych. Dla osiągnięcia efektu konieczne jest skopiowanie nieco ponad 32 kB, gdyż taki właśnie obszar jest na początku systemu plików JFS niewykorzystywany przez jego struktury:

```

1 [root@messy ~]# dd if=/dev/urandom of=/dev/sda7 bs=4096 count=9
2 9+0 records in
3 9+0 records out
4 36864 bytes (37 kB) copied, 0.0382422 s, 964 kB/s
5 [root@messy ~]# fsck.jfs -j /dev/sdc1 /dev/sda7
6 fsck.jfs version 1.1.14, 06-Apr-2009
7 processing started: 11/11/2009 15.25.35
8
9 /dev/sda7 is mounted.
10
11 WARNING!!!
12 Running fsck on a mounted file system
13 may cause SEVERE file system damage.
14
15 Do you really want to continue (y/n)? y
16 yes
17
18 Using default parameter: -p
19 The current device is: /dev/sda7
20 Block size in bytes: 4096
21 Filesystem size in blocks: 132528
22 **Phase 0 - Replay Journal Log
23 logredo failed (rc=273). fsck continuing.
24 **Phase 1 - Check Blocks, Files/Directories, and Directory Entries
25 **Phase 2 - Count links
26 **Phase 3 - Duplicate Block Rescan and Directory Connectedness
27 **Phase 4 - Report Problems
28 **Phase 5 - Check Connectivity
29 **Phase 6 - Perform Approved Corrections
30 **Phase 7 - Rebuild File/Directory Allocation Maps
31 **Phase 8 - Rebuild Disk Allocation Maps
32 530112 kilobytes total disk space.
33 3147 kilobytes in 647 directories.
34 98910 kilobytes in 12997 user files.
35 0 kilobytes in extended attributes
36 7557 kilobytes reserved for system use.
37 426792 kilobytes are available for use.
38 Filesystem is clean.

```

Jak widać z powyższego raportu, program *fsck.jfs* rozpoczyna pracę od otwarcia zestawu plików, przeczytania jego bloku identyfikacyjnego i określeniu na podstawie jego zawartości wartości podstawowych atrybutów, jak np. rozmiar bloku danych czy całkowity rozmiar zestawu (linie 20, 21). Następnie podejmuje próbę odtworzenia transakcji zatwierdzonych do wykonania w dzienniku, a niewprowadzonych w całości w zestawie plików (linie 22, 23). Jak widać krok ten zakończył się niepowodzeniem. Program *fsck* uruchamia procedurę naprawy składającą się z ośmiu etapów, w których wykonywane są następujące czynności:

1. W pierwszym etapie sprawdzane są wszystkie przydzielone plikom i katalogom bloki danych. Program kontroluje, czy struktury opisujące pliki oraz wpisy w katalogach zawierają wskazania do poprawnych obszarów kompleksu.
2. Drugi etap polega na wyznaczeniu liczby dowiązań do każdego pliku i katalogu dla sprawdzenia, czy wartość licznika dowiązań przechowywana w strukturach opisujących plik lub katalog jest poprawna.
3. W etapie trzecim program *fsck.jfs* przeszukuje kompleks w poszukiwaniu bloków danych zawłaszczonych przez więcej niż jeden plik lub katalog. Równocześnie poddaje weryfikacji strukturę B+ drzewa przechowującego informacje o położeniu katalogów w zestawach plików.
4. Wypisanie raportu o stwierdzonych niespójnościach ma miejsce w etapie czwartym. Do tego momentu nie zostały jeszcze podjęte żadne próby naprawy.
5. W etapie piątym sprawdzane jest, czy każdy obiekt kompleksu ma reprezentujący go węzeł w B+ drzewie struktury katalogów.
6. Program *fsck.jfs* podejmuje próbę usunięcia wszystkich błędów, o których raporty pojawiły się w etapie czwartym.
7. W etapie siódmym, na podstawie utworzonych list bloków danych i obszarów przydzielonych plikom i katalogom, weryfikowana jest poprawność map bitowych zestawów plików, czyli struktur przechowujących informacje o wykorzystaniu ich zasobów.
8. W ostatnim, ósmym etapie weryfikacji podlegają mapy bitowe przechowujące informacje o wykorzystaniu bloków dyskowych całego kompleksu z uwzględnieniem bloków przechowujących metadane samego systemu plików.

Po usunięciu błędów, lub jeśli system plików nie wymagał naprawy, program *fsck.jfs* oznacza go jako *czysty* i kończy działanie. Jeśli natomiast błędów nie udało się usunąć konieczne staje się użycie programów narzędziowych działających na niższym poziomie, jak np. edytora systemu plików *jfs\_debugfs*.

Program *fsck.jfs*, oprócz omówionych opcji **-p** oraz **-j**, posiada inne spośród których do najczęściej wykorzystywanych należą:

- **-a** – opcja wymusza działanie programu *fsck.jfs* w trybie autokorekty. Polega on na wprowadzeniu transakcji zapisanych w dzienniku. Jeśli jednak nie będzie on zaznaczony jako „czysty” program zakończy działanie bez wykonywania dalszych etapów. Opcja ta jest domyślną przy uruchamianiu programu podczas inicjalizacji systemu operacyjnego.
- **-f** – użycie opcji spowoduje wprowadzenie transakcji z dziennika oraz weryfikację spójności systemu plików, jeśli flaga stanu będzie wskazywała, że jest on „czysty”. Napotkane niespójności będą usuwane automatycznie.

- **-n** – opcja umożliwia działanie programu w trybie tylko do odczytu. Transakcje zapisane w dzienniku nie zostaną wprowadzone do systemu plików, a o stwierdzonych błędach pojawi się tylko informacja – nie zostanie podjęta próba ich naprawy.
- **-v** – tryb gadatliwy, dostarczający większej ilości informacji o znalezionych niespójnościach.
- **-omit\_journal\_replay** – opcja spowoduje, że program *fsck.jfs* przystąpi do usuwania niespójności systemu plików z pominięciem odtwarzania transakcji zapisanych w dzienniku.
- **-replay\_journal\_only** – usuwanie niespójności zostanie ograniczone jedynie do wprowadzenia transakcji, o których informacje znajdują się w dzienniku.

### Wybrane programy narzędziowe

Do najczęściej używanych programów narzędziowych dla systemu plików JFS należą:

- *jfs\_debugfs* – to niskopoziomowy edytor dla systemu plików JFS. Uruchomienie wymaga podania jako argumentu nazwy urządzenia, na którym znajduje się system plików. Program posiada własną linię komend. Przykład użycia dla wypisania dostępnych komend oraz wynik działania komendy *fsckw*, służącej wypisaniu nagłówka kompleksu, przedstawiono poniżej.

```

1 [root@messy ~]# jfs_debugfs /dev/sda7
2 jfs_debugfs version 1.1.14, 06-Apr-2009
3
4 Aggregate Block Size: 4096
5
6 > help
7
8             jfs_debugfs Commands
9
10 a[lter] <block> <offset> <hex string>
11 b[tree] <block> [<offset>]
12 cb[blfsck]
13 dir[ectory] <inode number> [<fileset>]
14 d[isplay] [<block> [<offset> [<format> [<count>]]]]
15 dm[ap]
16 dt[ree] <inode number> [<fileset>]
17 fsckw[sphdr]
18 h[elp] [<command>]
19 ia[g] [<IAG number>] [a | s | <fileset>]
20 i[node] [<inode number>] [a | s | <fileset>]
21 logs[uper]
22 q[uit]
23 se[t] [<variable> <value>]
24 su[perblock] [p | s]
25 s2p[erblock] [p | s]
26 u[nset] <variable>
27 xt[ree] <inode number> [<fileset>]
28
29 > fsckw
30 [1] eyecatcher:                'wspblkmp'
    [2] last_entry_pos:          0

```

```

31 [3] next_entry_pos:      0
32 [4] start_time:         11/11/2009 15:25:41
33 [5] end_time:           11/11/2009 15:25:46
34 [6] return_code:        0
35 [7] super_buff_addr:    0x80afaa0
36 [8] agg_record_addr:    0x0x80aa4c0
37 [9] bmap_record_addr:   0x0x80af500
38 [10] fscklog_agg_offset: 0x000000002057e000 (0d0542629888)
39 [11] fscklog_full:      0
40 [12] fscklog_buf_allocated: -1
41 [13] fscklog_buf_alloc_err: 0
42 [14] num_logwrite_errors: 0
43 display_fsck_wsphdr: [m]odify or e[x]it:

```

- *jfs\_fscklog* – program umożliwia dostęp do dzienników tworzonych przez program *fsck.jfs*. Służy także ich formatowaniu oraz wypisywaniu. Opcja **-e** umożliwia podanie nazwy urządzenia, którego dziennik powstał podczas ostatniego uruchomienia komendy *fsck.jfs* zostanie udostępniony i zapisany do pliku. Dla ostatnio wykonanego, testowego uszkodzenia systemu plików, dziennik jest następujący:

```

1 [root@messy ~]# jfs_fscklog -e /dev/sda7
2 jfs_fscklog version 1.1.14, 06-Apr-2009
3 Invalid magic number in the superblock (P). [extract.c:733]
4 Primary superblock is corrupt. [extract.c:953]
5 XCHKLOG S superblock is valid. [extract.c:979]
6 XCHKLOG Most recent fsck service log extracted into: fscklog.new [extract.c:552]
7 XCHKLOG fsck service log selected: MOST RECENT [extract.c:468]

```

Jak widać, uszkodzeniu uległ podstawowy blok opisu. Raport ten został także zapisany do pliku o nazwie *fscklog.new* znajdującego się w katalogu bieżącym. Format tego pliku jest formatem wewnętrznym, stosowanym przez program narzędziowy *jfs\_fscklog*. Wypisanie jego zawartości wymaga użycia opcji **-f** z wartością będącą nazwą pliku.

- *jfs\_logdump* – program umożliwia zapisanie zawartości dziennika systemu plików JFS do zewnętrznego pliku tekstowego. Plik ten nosi nazwę *jfslog.dmp*. Jego fragment przedstawiono poniżej:

```

1 JOURNAL SUPERBLOCK:
2 -----
3 magic number: x 87654321
4 version      : x 1
5 serial       : x 1
6 size         : t 10040 pages (4096 bytes/page)
7 bsize        : t 4096 bytes/block
8 l2bsize      : t 12
9 flag         : x 10200100
10 state        : x 0
11 end          : x 202c
12

```

### 3.4. ADMINISTRACJA WYBRANYMI SYSTEMAMI PLIKÓW W DYSTRYBUCJI REDHAT247

```

13  =====
14
15  logrec d 0    Logaddr= x 11147bc    Nextaddr= x 1114798    Backchain = x 1114798
16
17  LOG_COMMIT    (type = d 32768)    logtid = d 26720    aggregate = d 0
18
19      data length = d 0
20
21  -----
22
23  logrec d 1    Logaddr= x 1114798    Nextaddr= x 11146f0    Backchain = x 11146f0
24
25  LOG_REDOPAGE    (type = d 2048)    logtid = d 26720    aggregate = d 0
26
27      data length = d 132    fileset = d 1    inode = d 16 (x 10)
28      type = d 1 REDOPAGE:INODE
29      l2linesize = d 7    pxd length = d 1    phys offset = x c8bb (d 51387)
30
31      0x8104320    37AFF94A    10000000    3A640000    7A350000    7..J....:d..z5..
32      0x8104330    04000000    B8C80000    00200000    00000000    .....
33      0x8104340    02000000    00000000    01000000    00000000    .....
34      0x8104350    00000000    A4810600    A5C8FA4A    DACCA12B    .....J...+
35      0x8104360    A5C8FA4A    46CBF22B    A5C8FA4A    46CBF22B    ...JF...+...JF...+
36      0x8104370    A5C8FA4A    00000000    00000000    00000000    ...J.....
37      0x8104380    00000000    00000000    00000000    00000000    .....
38      *
39      0x81043a0    08000100    ....
40
41  -----
42  .....

```

Program ten to klasyczne narzędzie niskopoziomowe. Dostarczoną informację można przykładowo wykorzystać do „ręcznej” korekty systemu plików przy pomocy programów *jfs\_debugfs*, gdy inne programy, służące jego automatycznej weryfikacji zawiodą.

- *jfs\_tune* – to program narzędziowy umożliwiający wypisanie oraz zmianę wartości podstawowych parametrów linuksowego systemu plików JFS. Opcja **-l** umożliwia wypisanie informacji o systemie plików. Jako argument wywołania komendy należy podać ścieżkę dostępu do pliku urządzenia, na którym dany system się znajduje. Stąd mamy:

```

1  [root@messy ~]# jfs_tune -l /dev/sda7
2  jfs_tune version 1.1.14, 06-Apr-2009
3
4  JFS filesystem superblock:
5
6  JFS magic number:      'JFS1'
7  JFS version:          2
8  JFS state:            clean
9  JFS flags:            JFS_LINUX  JFS_COMMIT  JFS_GROUPCOMMIT

```

```

10 Aggregate block size: 4096 bytes
11 Aggregate size: 1059776 blocks
12 Physical block size: 512 bytes
13 Allocation group size: 8192 aggregate blocks
14 Log device number: 0x821
15 Filesystem creation: Tue Nov 10 19:21:43 2009
16 File system UUID: 403f3dcc-8d5e-40ab-99b0-b8d0969e112e
17 Volume label: ''
18 External log UUID: 4275729f-1bec-48e5-a8d9-ae20df920eff

```

Spśród innych opcji programu należy wskazać na opcję **-L**. Jej wartością jest etykieta, która zostanie nadana wskazanemu systemowi plików. Jednak jak pamiętamy, używanie etykiet do rozpoznawania systemów plików może w niektórych przypadkach prowadzić do niejednoznaczności. Stąd wprowadzono niepowtarzalny numer identyfikacyjny urządzenia. Można go nadać systemowi plików korzystając z opcji **-U**. Opcja ta wykorzystuje jeden z trzech specyfikatorów:

1. *clear* – spowoduje on usunięcie istniejącego identyfikatora,
2. *random* – wygeneruje oraz przypisze identyfikator o wartości losowej,
3. *time* – wygeneruje oraz przypisze identyfikator bazujący na bieżącym czasie.

Ostatnią z istotnych opcji jest opcja **-J**. Umożliwia ona skojarzenie utworzonego komendą *mkfs.jfs* dziennika z istniejącym systemem plików. Jako wartość opcji po specyfikatorze *device*= podaje się nazwę pliku urządzenia, na którym znajduje się dziennik, zaś argumentem jest nazwa pliku urządzenia, na którym założono system plików JFS. Przykładowo:

```

1 [root@messy ~]# jfs_tune -J device=/dev/sdc1 /dev/sda7
2 jfs_tune version 1.1.14, 06-Apr-2009
3 Attached JFS external journal to JFS FS successfully.

```

### 3.4.7 System plików ReiserFS

System plików ReiserFS, był pierwszym systemem wyposażonym w mechanizm dziennikowania, przeznaczonym dla systemów linuksowych. Zastosowane rozwiązania konstrukcyjne i implementacyjne spowodowały, że wyróżniał się on na tle innych systemów plików stabilnością oraz czasem dostępu do plików charakteryzujących się dużymi rozmiarami. Twórcą systemu jest Hans Reiser, absolwent Uniwersytetu Kalifornijskiego w Berkeley. Twierdził on, że systemy plików wykorzystywane w systemach uniksowych zatrzymały się w swym rozwoju w latach siedemdziesiątych dwudziestego wieku. Dla poprawy tego stanu rzeczy założył on firmę *Namesys*, której głównym celem miało być konstruowanie nowoczesnych systemów plików. Firma otrzymała liczne dotacje. Sponsorowała ją m.in. DARPA, firma Linspire (LindowOS) producent systemów Debian GNU/Linux, a później Ubuntu, oraz BigStorage i ApplianceWare – producenci pamięci masowych. Ostatnim produktem firmy *Namesys* jest system plików ReiserFS w wersji 3.6. Po jego opublikowaniu przystąpiono do tworzenia zupełnie nowego systemu o nazwie Reiser4. Jednak 10 października 2006 roku Hans Reiser został aresztowany w związku z podejrzeniem o zamordowanie swojej żony. W grudniu 2006 roku zdecydował o sprzedaży firmy, a uzyskane pieniądze przeznaczył na pokrycie kosztów procesu. 28 kwietnia 2008 roku został uznany winnym i skazany na dożywotnie więzienie. Wydarzenia te mocno zachwiały pozycją systemów plików ReiserFS. W roku 2006 przestał on być wykorzystywany jako podstawowy w systemach operacyjnych S.u.S.E.,

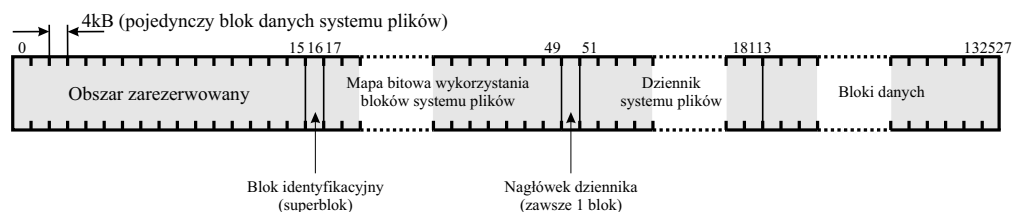


a jego miejsce zajął ext3. Jądra systemu Linux wspierają ReiserFS do wersji 3.6. Od stycznia 2008 roku, czyli od chwili zakończenia działalności przez firmę Namesys, prace nad rozwojem systemu plików Reiser4 są prowadzone pod kierunkiem jej byłego pracownika Edwarda Szyszki. Na jego stronie www ([http://chichkin\\_i.zelnet.ru/namesys](http://chichkin_i.zelnet.ru/namesys)) można zleżeć ostatnie wersje programów narzędziowych oraz łat na jądra systemu Linux 2.6 dla systemu plików Reiser4.

W kolejnych podrozdziałach omówiona zostanie zarówno wersja 3.6 systemu ReiserFS jak i Reiser4.

### Budowa wewnętrzna

**ReiserFS 3.6** Partycja, na której znajduje się system plików ReiserFS 3.6 jest podzielona na bloki o jednakowym rozmiarze. W systemach linuksowych rozmiar domyślny wynosi 4096 B. Teoretycznie maksymalnym rozmiarem jest 65536 B, gdyż odpowiedni atrybut w bloku identyfikacyjnym wykorzystuje do przechowywania tej wartości 2 bajty. Rozmiar minimalny to 512 B. Bloki są ponumerowane, od 0 począwszy. Maksymalna liczba bloków wynosi  $2^{32}$ . Widać zatem, że maksymalny rozmiar systemu plików może teoretycznie wynosić  $2^{32} \times 65536$  B czyli 256 TB.



Rysunek 3.17: Schemat struktury logicznej systemu plików Reiserfs w wersji 3.6.

Schemat budowy systemu plików, przedstawiony na rysunku 3.17, został sporządzony dla bloku o rozmiarze 4096 B. Jak widać system plików rozpoczyna nie wykorzystywany przez system plików obszar o rozmiarze 64 kB. Może on być pomocny dla programów ładujących system operacyjny lub do przechowywania etykiety partycji. Bezpośrednio za nim znajduje się blok identyfikacyjny (superblok) systemu plików. Zawiera on wartości atrybutów opisujących system plików. Mimo iż jego rozmiar wynosi 80 B, zajmuje on cały blok. Blok identyfikacyjny systemu plików przechowuje informacje o:

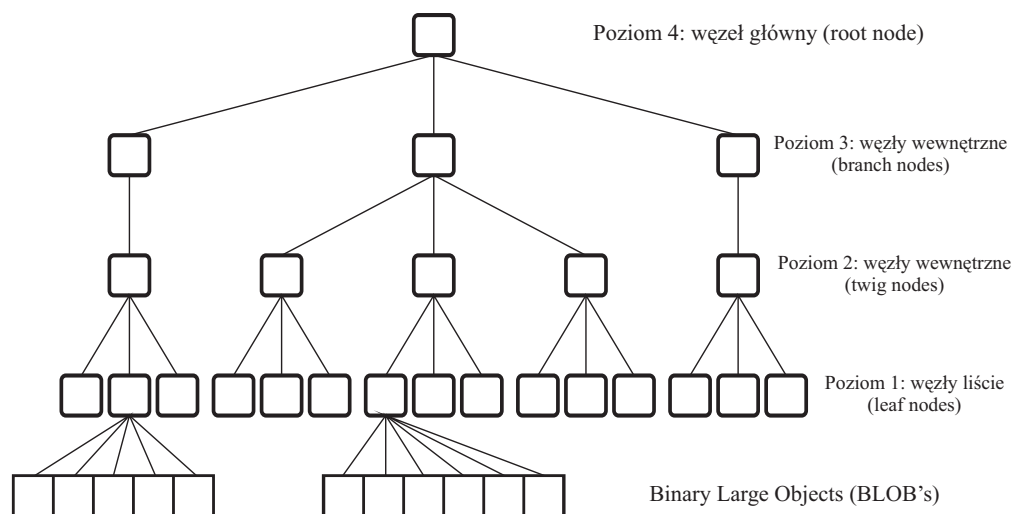
- całkowitej liczbie bloków partycji - 4 B,
- liczbie wolnych bloków partycji - 4 B,
- numerze bloku przechowującego katalog główny systemu plików - 4 B,
- numerze bloku przechowującego pierwszy węzeł dziennika.
- numerze urządzenia na którym znajduje się dziennik systemu plików - 4 B,
- aktualnym rozmiarze dziennika. Atrybut jest wykorzystywany jeśli w systemie istnieje możliwość utworzenia dziennika o rozmiarze innym niż domyślny - 4 B,
- maksymalnej liczbie bloków służących opisowi pojedynczej transakcji przechowywanej w dzienniku - 4 B,
- wartości losowej dziennika - 4 B,

- maksymalnej liczbie transakcji przechowywanych w dzienniku - 4 B,
- maksymalnym wieku (w sekundach) transakcji asynchronicznej przechowywanej w dzienniku, która musi zostać odwzorowana w systemie plików - 4 B,
- maksymalnym wieku transakcji przechowywanej w dzienniku - 4 B,
- rozmiarze bloku systemu plików w bajtach - 2 B,
- maksymalnym rozmiarze tablicy identyfikatorów obiektu - 2 B,
- bieżącym rozmiarze tablicy identyfikatorów obiektu - 2 B,
- statusie partycji (1 - poprawny, 2 - błędny) - 2 B. Może zdarzyć się, że stan zamontowanego systemu plików będzie równy 2, jeśli został on ręcznie zamontowany po awarii systemu. Właściwy status zostanie zapisany podczas kolejnej inicjalizacji systemu,
- „magicznym” napisie (powinien być: *ReIsEr2Fs*) - 12 B,
- identyfikatorze funkcji mieszającej, stosowanej do umieszczania w/g nazw plików w katalogach - 4 B,
- aktualnej głębokości B+ drzewa indeksującego informację przechowywaną w systemie plików - 2 B,
- liczbie bloków przechowujących mapę bitową zajętości bloków systemu plików - 2 B,
- numerze wersji systemu plików - 2 B. W praktyce jest to numer korekty (*revision*) struktury systemu plików,
- nieużywanym polu - 2 B,
- numerze ostatnio utworzonego i-węzła - 4 B. Wartość ta jest korygowana po wykonaniu operacji równoważenia B+ drzewa.

Strukturą wykorzystywaną do przechowywania danych w systemie plików jest zbalansowane drzewo, określane w dokumentacji jako B+ lub S+. Drzewo tworzą tzw. węzły. Odpowiadają one blokom dyskowym. Węzły, które nie zawierają metadanych to węzły niesformatowane. Węzły, które składają się z listy wpisów, to węzły sformatowane. Pozostałe, to węzły wewnętrzne. Węzły niesformatowane znajdują się na najniższym poziomie drzewa, węzły sformatowane na poziomie pierwszym zaś na pozostałych poziomach występują węzły wewnętrzne. Strukturę drzewa przedstawiono na rysunku 3.18

Informacja przechowywana jest w systemie plików w postaci wpisów. Co najmniej jeden wpis jest przechowywany w każdym węźle sformatowanym. Reprezentuje on obiekt przechowywany w systemie plików (plik, katalog, wpis informacyjny). Wpis ma przypisany unikalny numer (zwany kluczem), który odpowiada numerowi i-węzła w innych systemach plików. Węzeł wewnętrzny zawiera klucze oraz wskaźniki do jego węzłów potomnych. Pierwszy wskaźnik (umownie nazwany P0) wskazuje na obiekty posiadające identyfikatory o wartości mniejszej niż wartość odpowiadającego mu pierwszego identyfikatora (K0). Drugi wskaźnik, P1, wskazuje na obiekty o wartości identyfikatora K spełniającego warunek  $K0 \leq K < K1$ . Ostatni wskaźnik wskazuje obiekty o wartości klucza większej od wartości ostatniego klucza w danym węźle. Uporządkowanie to znacznie przyspiesza wstawianie, wyszukiwanie i usuwanie obiektów.

Wpisy służą do przechowywania danych zapisywanych w systemie plików. Rozróżnia się cztery rodzaje wpisów. Są to wpisy informacyjne (*Stat Item*), wpisy katalogowe (*Directory Item*), wpisy bezpośrednie (*Direct Item*) i wpisy pośrednie (*Indirect Item*).



Rysunek 3.18: Struktura drzewa systemu plików Reiserfs w wersji 3.6.

Wpisy informacyjne w systemie ReiserFS 3.6 mają długość 44 B i zawierają następującą informację:

- typ pliku i prawa dostępu - 2B,
- pole zarezerwowane - 2B,
- liczba dowiązań twardych - 4B,
- rozmiar pliku w bajtach - 8B,
- numer identyfikacyjny właściciela indywidualnego pliku - 4B,
- numer identyfikacyjny właściciela grupowego pliku - 4B,
- czas ostatniego dostępu do pliku - 4B,
- czas ostatniej modyfikacji pliku - 4B,
- czas ostatniej modyfikacji danych opisujących plik - 4B,
- liczbę bloków dyskowych wykorzystywanych przez plik - 4B,
- numer urządzenia (jeśli wpis dotyczy pliku reprezentującego urządzenie fizyczne) lub numer i-węzła dla wpisu pliku, katalogu lub dowiązania symbolicznego.

Wpisy katalogowe zawierają listę obiektów znajdujących się w katalogu. Wpis składa się ze zbioru nagłówek oraz listy nazw plików. Jeśli liczba plików znajdujących się w katalogu jest na tyle duża, że lista ich nazw nie mieści się jednym wpisem, to wówczas może ona obejmować kilka wpisów, a stosowna informacja zostaje zapisana w nagłówkach wpisu.

Wpisy bezpośrednie przechowują dane małych plików (o rozmiarach mniejszych od rozmiaru węzła) lub końcówki większych plików. Dane dużych plików przechowywane są w węzłach nazywanych *Binary Large Objects* (BLOB). Wpisy pośrednie zawierają listę wskaźników do węzłów niesformatowanych.

W systemie plików ReiserFS w wersji 3.6 zrealizowano tzw. *dziennikowanie w ustalonym miejscu*. Dziennik został zaprojektowany jako ciągła przestrzeń bloków dyskowych. Jego rozmiar zależy od wersji jądra systemu operacyjnego. Domyślnie są to 8192 bloki dyskowe przeznaczone do przechowywania listy transakcji oraz jeden blok zawierający nagłówek dziennika, który znajduje się na jego końcu. Transakcja może zostać zapisana na maksymalnie trzech blokach danych. Dziennik pracuje jak bufor cykliczny. Jeśli wykorzystany zostanie ostatni jego blok, kolejne transakcje zapisywane są od bloku pierwszego. Dziennikowanie dotyczy jedynie metadanych systemu plików.

Mimo, iż dziennik zajmuje cały blok danych, to przechowywana w nim informacja zapisana jest na 12 bajtach. Informacja ta składa się z numeru identyfikacyjnego ostatniej odwzorowanej na systemie plików transakcji (4 B), odległości liczonej w blokach od początku systemu do bloku zawierającego opis następnej transakcji zapisanej w dzienniku (4 B) oraz numeru identyfikacyjnego aktualnie odwzorowywanej transakcji (*mount ID*).

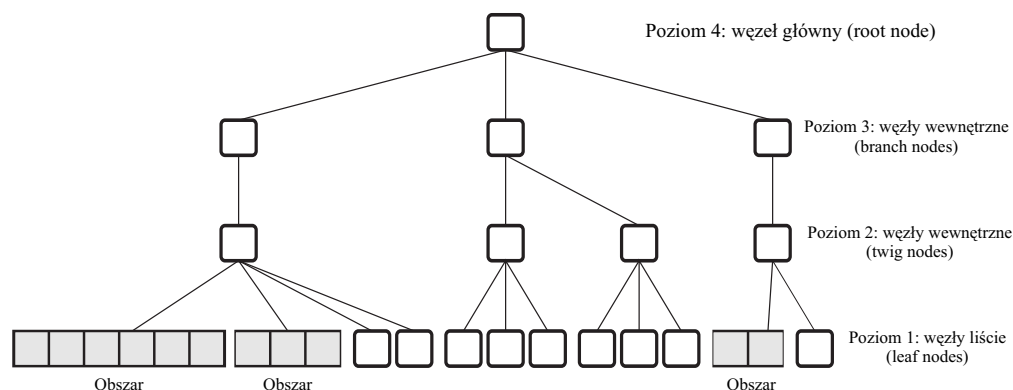
Transakcja zawiera opis zmian w metadanych systemu plików. Jej implementacja składa się z bloku opisu transakcji, listy bloków zawierających związane z nią dane i bloku zamykającego. Blok opisu zawiera numery identyfikacyjne transakcji (4 B) i montowania (4 B), liczbę bloków zajmowanych przez transakcję (4 B), bloki transakcji oraz tzw. liczbę magiczną (*magic number*), którą jest napis *ReIsErLB* (12 B). Miejsce zajmowane przez bloki transakcji jest teoretycznie zależne od rozmiaru bloku systemu plików. Skoro pozostałe wartości wchodzące w skład bloku opisu transakcji mają ustalony rozmiar wynoszący łącznie 24 B, to opis transakcji zajmuje miejsce równe rozmiarowi bloku pomniejszonemu o 24 B.

**Reiser4** Najistotniejszą zmianą jaka pojawiła się w systemie plików było wprowadzenie znanych z innych systemów plików ekstentów, czyli obszarów przylegających do siebie bloków danych, które mogą być przydzielane do przechowywania danych jednego pliku. Z punktu widzenia struktur danych systemu plików Reiser4, w ekstencie przechowywane są węzły niesformatowane. Wskaźnik do ekstentu zawiera numer bloku danych rozpoczynającego obszar oraz jego długość (liczbę tworzących go bloków). Z przedstawionego na rysunku 3.19 schematu widać, iż można w ten sposób uzyskać zwiększenie liczby rozgałęzień drzewa. Jednocześnie zapisanie informacji o rozmieszczeniu pliku na dysku wymaga użycia mniejszej liczby metadanych. Dodatkowo, drzewo może być bardziej zrównoważone, gdyż ekstenty zajmują w nim ten sam poziom co liście. Reasumując, wprowadzenie ekstentów pozwoliło zwiększyć wydajność systemu plików przez skrócenie średniej drogi od korzenia do węzła na najniższym poziomie. Ograniczyło również liczbę węzłów wewnętrznych, a więc samych metadanych i operacji na nich.

W systemie plików Reiser4 nowe bloki dla przechowywania danych są alokowane dopiero wówczas, gdy bufor dyskowy staje się pełny i zapisanie ich zawartości na dysku staje się konieczne. Wykonanie funkcji systemowej *write* skutkuje jedynie zapisem do buforów. Opóźniony zapis daje czas do działania zaawansowanym algorytmom wyszukiwania bloków dyskowych, co pozwala na tworzenie ekstentów o odpowiednich rozmiarach oraz zapewnia, że wykorzystywane przez ten sam plik będą znajdowały się na dysku w bliskim sąsiedztwie.

Usunięcie wpisu skutkuje zwolnieniem miejsca na dysku jedynie wówczas, gdy jest to ostatni wpis w węźle. Proces pakowania wpisów jest wykonywany niejako „przy okazji”, np. aktualizacji danych na dysku. Ta z kolei jest wykonywana, jeśli bufor jest wypełniony co gwarantuje, że znajduje się w nim wiele wpisów. Pozwala to na ich efektywne upakowanie oraz ogranicza późniejsze operacje odczytu. Różnice w zapisywaniu i usuwaniu bloków powodują, że organizacja danych w systemie plików Reiser4 nie można nazwać B+ drzewem. Stąd nazwa *tańczące drzewo* (*dancing tree*).

Dziennikowanie w systemie plików Reiser4 dotyczy danych oraz metadanych. Spowodowało to zmniejszenie efektywności systemu, ale wprowadzenie ekstentów z nawiązką je zrekompensowało.



Rysunek 3.19: Struktura drzewa systemu plików Reiser4.

Jednak najistotniejsza zmiana to zastosowanie techniki zaproponowanej przez Dave Hitza o nazwie *Write Anywhere File Layout system* (WAFL)<sup>5</sup>. Polega ona na tym, że wykonanie operacji przykładowo zapisu do pliku, powoduje zapisanie danych w pewnym miejscu na dysku. Dane te nie są zapisane bezpośrednio w drzewie. Są w nim zapisane jedynie informacje, które wskazują na miejsce zapisu danych. Wymusza to konieczność uaktualniania wskaźników. Takie podejście zapewnia całkowitą atomowość wszystkich operacji w systemie plików nie powodując jednocześnie narzutu związanego z zapisywaniem wszystkich zmian dwa razy – najpierw w dzienniku, a następnie utrwalania zmian w systemie plików.

### Zakładanie systemu plików

Programy narzędziowe dla systemów plików ReiserFS oraz Reiser4 są dostępne w postaci kodów źródłowych odpowiednio w pakietach *reiserfsprogs* oraz *reiser4progs*. Kompilacja (polecenia `./configure` oraz `make`) doprowadzi do stworzenia programów, zaś w wyniku instalacji (polecenie `make install`) zostaną one skopiowane do katalogu `/usr/local/sbin`.

**ReiserFS 3.6** Program służący do zakładania systemu plików ReiserFS w wersji 3.6 jest zapisany w pliku o nazwie *mkreiserfs*. Rozwiązanie to nie spełnia zasady zgodnie z którą program do tworzenia systemu plików określonego typu powinien znajdować się w pliku o nazwie *mkfs.<typ>*. W takim przypadku jest on uruchomiony przez program *mkfs* użyty z opcją `-t typ`. Rozwiązaniem może być np. utworzenie w katalogu `/sbin` dowiązań symbolicznych o nazwach *mkfs.reiserfs* wskazujących na plik `/usr/local/sbin/mkreiserfs`:

```
[root@messy sbin]# ln -s /usr/local/sbin/mkreiserfs ./mkfs.reiserfs
```

Utworzenie dowiązania symbolicznego, przedstawionego powyżej umożliwi zakładanie systemów plików ReiserFS z wykorzystaniem programu *mkfs*, uruchomionego z podaniem typu pliku oraz nazwy pliku urządzenia na którym system ten ma zostać założony. Rozpocznijmy od systemu plików ReiserFS 3.6:

<sup>5</sup>Dla przypomnienia, istnieją dwa inne sposoby umieszczania dziennika systemu plików. Najprostsze to dziennik jako jeden z plików w systemie plików (stosowane np. w systemie plików ext3). Rozwiązanie to charakteryzuje się niską wydajnością i słabą odpornością na awarię. Rozwiązanie alternatywne polega na umieszczeniu dziennika w specjalnym, wydzielonym obszarze (system plików XFS) dzięki czemu istnieje możliwość optymalizowania dostępu oraz zapewnienia większego bezpieczeństwa dzięki odseparowaniu go od zwykłych plików.

```
1 [root@messy sbin]# mkfs -t reiserfs -d /dev/sda7
2 mkfs.reiserfs 3.6.21 (2009 www.namesys.com)
3
4 A pair of credits:
5 Oleg Drokin was the debugger for V3 during most of the time that V4 was under
6 development, and was quite skilled and fast at it. He wrote the large write
7 optimization of V3.
8
9 Elena Gryaznova performed testing and benchmarking.
10
11 Guessing about desired format.. Kernel 2.6.27.1 is running.
12 Block 16 (0x807) contains super block. Format 3.6 with standard journal
13 Count of blocks on the device: 132528
14 Number of blocks consumed by mkreiserfs formatting process: 8216
15 Free blocks: 124312
16 Blocksize: 4096
17 Hash function used to sort names: "r5"
18 Number of bitmaps: 5
19 Root block: 8211
20 Tree height: 2
21 Objectid map size 2, max 972
22 Journal parameters:
23     Device [0x0]
24     Magic [0x59d3471f]
25     Size 8193 blocks (including 1 for journal header) (first block 18)
26     Max transaction length 1024 blocks
27     Max batch size 900 blocks
28     Max commit age 30
29 Filesystem state 0x0
30 sb_version 2
31 inode generation number: 0
32 UUID: a59712a3-159b-40b4-ae3c-4ca17e3eb5fd
33 ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
34     ALL DATA WILL BE LOST ON '/dev/sda7'!
35 Continue (y/n):y
36 Initializing journal - 0%....20%....40%....60%....80%....100%
37 Syncing..ok
```

W wywołaniu polecenia użyto dodatkowo opcję *-d*. Służy ona do wypisywania precyzyjnych informacji dotyczących wartości atrybutów zakładanego systemu plików. Istotne informacje zaczynają się od linii 12. Dotyczą one wersji aktualnie pracującego jądra systemu operacyjnego. Kolejne informacje dotyczą już zakładanego systemu plików. Zauważmy, że blok identyfikacyjny systemu plików znajduje się w bloku 16. Całkowita liczba bloków systemu plików wynosi 132528, z czego dostępnych dla przechowywania danych jest 128312. Rozmiar bloku danych wynosi 4096 B. Dziennik został założony na tym samym urządzeniu co system plików. Jego rozmiar wynosi 8193 bloki. Jedna transakcja może wykorzystywać do 1024 bloków.

Program *mkreiserfs* posiada kilka opcji, które umożliwiają zmianę domyślnych wartości atrybutów zakładanego systemu plików. Te dotyczące dziennika zostaną omówione w następnym

punkcie. Spośród pozostałych należy wymienić:

- **-b** *rozmiar\_bloku\_danych* – opcja pozwala ustalić rozmiar bloku danych systemu plików podawany w bajtach. Obsługiwane wartości stanowią potęgę liczby 2 z przedziału 512–8192.
- **-h** *funkcja\_mieszająca* – wartość opcji określa algorytm mieszający wykorzystywany do wyszukiwania i zapisywania plików w katalogu. W systemie plików ReiserFS zaimplementowane zostały następujące funkcje mieszające:
  - *rupasov* – to prosty, a zarazem wydajny algorytm mieszający, który leksykalnie podobne nazwy plików umieszcza obok siebie. Powoduje to zwiększenie prawdopodobieństwa kolizji przy wyszukiwaniu miejsca dla kolejnej, podobnej nazwy.
  - *tea* – to implementacja funkcji mieszającej Davisa–Meyera, która klucze mieszające tworzy permutując bity ciągu znaków będącego nazwą pliku. Zmniejsza to prawdopodobieństwo kolizji wydłużając jednak czas generacji w stosunku do innych funkcji. Jest zalecana dla systemów plików przechowujących w katalogach duże ilości plików.
  - *r5* – funkcja ta stanowi modyfikację funkcji *rupasov* zapewniając rozsądny kompromis między niskim prawdopodobieństwem kolizji, a czasem obliczeń. Jest stosowana domyślnie.
  - Parametr *detect* – nie oznacza żadnej funkcji mieszającej. Wymusi jedynie na poleceniu *mount* analizę systemu plików i określenie stosowanego w systemie plików typu funkcji mieszającej.
- **-format** *wersja* – opcja pozwala wybrać format (wersję) systemu plików. Dla jądra w wersji 2.2 jedynym obsługiwanym to 3.5. Od jądra w wersji 2.4 włącznie obsługiwany jest także format 3.6. Wartością domyślną jest 3.6. Należy zwrócić uwagę iż wersje te nie są ze sobą kompatybilne. Dodatkowo, nie ma programów narzędziowych umożliwiających konwersję.
- **-u** *UUID* – wartością opcji jest uniwersalny numer identyfikacyjny urządzenia. Numer ten można wygenerować programem *uuidgen*. W przypadku zamiany systemów plików można wykorzystać używany dotychczas. Pomińcie opcji w wywołaniu komendy powoduje jego automatyczne wygenerowanie.
- **-l** *etykieta* – opcja umożliwia nadanie etykiety systemowi plików. Maksymalna długość to 16 znaków. Etykiety dłuższe zostaną obcięte przez program *mkreiserfs*.
- **-B** *plik\_bloków\_uszkodzonych* – opcja pozwala zdefiniować plik przechowujący listę uszkodzonych bloków systemu plików. Można ją wygenerować korzystając z komendy *badblocks*. Jej użycie wymaga podania rozmiaru bloku jako wartości opcji **-b** oraz jako argumentu ścieżki dostępu do pliku urządzenia. Dla utworzonego systemu plików ReiserFS postać komendy będzie następująca:

```
1 [root@messy ~]# badblocks -b 4096 /dev/sda7
```

**Reiser4** Program narzędziowy służący zakładaniu systemu plików Reiser4 znajduje się w pliku o nazwie *make\_reiser4*, który podobnie jak analogiczny program dla systemu ReiserFS 3.6, znajduje się w katalogu */usr/local/sbin*. Uruchamianie go przez program *mkfs* z opcją **-t** i wartością *reiser4* może zostać zrealizowana przez utworzenie dowiązania symbolicznego o nazwie *mkfs.reiser4* w katalogu */sbin*:

```
1 [root@messy sbin]# ln -s /usr/local/sbin/make_reiser4 ./mkfs.reiser4
```

Do założenia systemu plików Reiser4 wymagane jest podanie jedynie nazwy pliku reprezentującego urządzenie, na którym system ma zostać założony. Wartości pozostałych atrybutów oraz zestaw wtyczek, będą domyślne. Wtyczki zostały zaimplementowane jako moduły zawierające funkcje, które obsługują wybrane aspekty pracy systemu plików. Przykładowo o tym, która funkcja mieszająca będzie wykorzystywana decyduje wartość klucza *hash*. Wartość ta jest wskaźnikiem do funkcji. Uzyskanie informacji o wykorzystywanych modułach oraz wartościach innych atrybutów wymaga użycia opcji **-p**:

```
1 [root@messy ~]# mkfs -t reiser4 -p /dev/sda7
2 mkfs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by
4 reiser4progs/COPYING.
5
6 Default profile:
7 create:          "reg40"          (id:0x0 type:0x0)    [Regular file plugin for creat(2)]
8 key:             "key_large"      (id:0x1 type:0xb)    [Key plugin]
9 compress:        "lzo1"           (id:0x0 type:0xc)    [Compression plugin]
10 compressMode:    "conv"           (id:0x4 type:0xd)    [Compression Mode plugin]
11 cluster:         "64K"            (id:0x0 type:0x10)   [Cluster plugin]
12 hash:           "r5_hash"         (id:0x1 type:0x3)    [Directory entry hash plugin]
13 fibration:       "ext_1_fibre"    (id:0x2 type:0x4)    [Key fibration plugin]
14 formatting:      "smart"          (id:0x2 type:0x5)    [File body formatting plugin]
15
16 Block size 4096 will be used.
17 Linux 2.6.31.6 is detected.
18 Uuid 4275f577-3223-48f8-9f19-93a17c02aa88 will be used.
19 Reiser4 is going to be created on /dev/sda7.
20 (Yes/No): Yes
21 Creating reiser4 on /dev/sda7 ... done
```

Profil systemu plików, w skład którego wchodzi m.in. wtyczki, został opisany w liniach 7–14 powyższego listingu. Należą one zasadniczo do dwóch bibliotek. Pierwsza, per-Superblock Plugins Library (SPL) zawiera moduły pracujące na poziomie dysku. Obsługują one blok identyfikacyjny, i-węzły, mapy bitowe i-węzłów oraz dziennik. Drugą bibliotekę, per-File Plugins Library (FPL), tworzą tzw. zarządcy plików, a więc moduły odpowiedzialne za zarządzanie plikami oraz katalogami w systemie plików. U podstaw realizacji leżą klasy wirtualne i mechanizm późnego wiązania dostępne w każdym języku obiektowym (w tym przypadku w C++). Wtyczki znajdują się w bibliotece *libreiser4*. Niestety, w momencie powstawania niniejszego opracowania, nie był dostępny dokument zawierający opis wszystkich dostępnych wtyczek. Edward Szyszkini zapewniał jednak, że pojawi się on najpóźniej w 2010 roku.

Podczas zakładania systemu plików uwagę zwrócił zaskakująco krótki czas zakładania systemu plików. Zmierzony programem *time*, dla uruchomienia programu *mkfs.reiser4* z opcją **-y**, która powoduje, że program nie wymaga potwierdzenia operacji oraz nie wypisuje żadnych informacji, wynosił zaledwie 0.03 s podczas zakładania systemu plików na partycji o rozmiarze 0.5 GB.

Program *make.reiser4* nie posiada zbyt wielu opcji. Wynika to z faktu, że ciężar zmiany wartości atrybutów został przeniesiony na jedną opcję umożliwiającą manipulowanie wtyczkami.



Do podstawowych opcji, oprócz omówionych **-t** oraz **-y**, należą:

- **-f** – opcja wymusza założenie systemu plików na całym dysku, a nie tylko na urządzeniu blokowym lub zamontowanej partycji.
- **-b** *rozmiar\_bloku* – opcja umożliwia zdefiniowanie rozmiaru bloku danych wykorzystywanego w systemie plików. Wartość domyślna to rozmiar strony w danej architekturze.
- **-L** *etykieta* – wartością opcji jest etykieta, która zostanie nadana zakładanemu systemowi plików.
- **-U** *identyfikator* – opcja służy nadaniu unikalnego identyfikatora uniwersalnego.
- **-s** – użycie opcji spowoduje utworzenie katalogu *lost+found* w zakładanym systemie plików dla przechowywania osieroconych bloków danych czy katalogów katalogów. Katalog ten jest wykorzystywany m.in. przez program *fsck.reiser4*
- **-l** – opcja umożliwia wypisanie dostępnych w bibliotece *libreiser4* wtyczek, z których korzysta system plików. Poniżej przedstawiono ich listę:

```

1 [root@messy ~]# mkfs.reiser4 -l
2 mkfs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by
4 reiser4progs/COPYING.
5
6 Known plugins:
7 "reg40"          (id:0x0 type:0x0) [Unix-file regular file plugin.]
8 "dir40"          (id:0x1 type:0x0) [Directory file plugin.]
9 "sym40"          (id:0x2 type:0x0) [Symlink file plugin.]
10 "spl40"          (id:0x3 type:0x0) [Special file plugin.]
11 "ccreg40"        (id:0x4 type:0x0) [Cryptcompress regular file plugin.]
12 "stat40"         (id:0x0 type:0x1) [StatData item plugin.]
13 "cde40"          (id:0x2 type:0x1) [Compound directory entry item plugin.]
14 "nodeptr40"      (id:0x3 type:0x1) [Node pointer item plugin.]
15 "extent40"       (id:0x5 type:0x1) [Extent file body item plugin.]
16 "plain40"        (id:0x6 type:0x1) [Plain file body item plugin.]
17 "ctail40"        (id:0x7 type:0x1) [Compressed file body item plugin.]
18 "bbox40"         (id:0x8 type:0x1) [Safe link item plugin.]
19 "node40"         (id:0x0 type:0x2) [Node plugin.]
20 "rupasov_hash"   (id:0x0 type:0x3) [Rupasov hash plugin.]
21 "r5_hash"        (id:0x1 type:0x3) [R5 hash plugin.]
22 "tea_hash"       (id:0x2 type:0x3) [Tea hash plugin.]
23 "fnv1_hash"      (id:0x3 type:0x3) [Fnv1 hash plugin.]
24 "deg_hash"       (id:0x4 type:0x3) [Degenerate hash plugin.]
25 "lexic_fibre"    (id:0x0 type:0x4) [Lexicographic fibration plugin.]
26 "dot_o_fibre"    (id:0x1 type:0x4) ['o' fibration plugin.]
27 "ext_1_fibre"    (id:0x2 type:0x4) [1-symbol extention fibration plugin.]
28 "ext_3_fibre"    (id:0x3 type:0x4) [3-symbol extention fibration plugin.]
29 "extents"        (id:0x0 type:0x5) ['Extents only' tail policy plugin.]
30 "tails"          (id:0x1 type:0x5) ['Tails only' tail policy plugin.]
31 "smart"          (id:0x2 type:0x5) [Smart tail policy plugin.]

```

32	"sdext_lw"	(id:0x0 type:0x6)	[Light stat data extension plugin.]
33	"sdext_unix"	(id:0x1 type:0x6)	[Unix stat data extension plugin.]
34	"sdext_lt"	(id:0x2 type:0x6)	[Large times stat data extension plugin.]
35	"sdext_symlink"	(id:0x3 type:0x6)	[Symlink stat data extension plugin.]
36	"sdext_plugin_set"	(id:0x4 type:0x6)	[Plugin Set StatData extension plugin.]
37	"sdext_flags"	(id:0x5 type:0x6)	[Inode flags stat data extension plugin.]
38	"sdext_crypto"	(id:0x7 type:0x6)	[Crypto stat data extension plugin.]
39	"sdext_heir_set"	(id:0x8 type:0x6)	[Heir Set StatData extension plugin.]
40	"format40"	(id:0x0 type:0x7)	[Disk-format plugin.]
41	"oid40"	(id:0x0 type:0x8)	[Inode number allocator plugin.]
42	"alloc40"	(id:0x0 type:0x9)	[Space allocator plugin.]
43	"journal40"	(id:0x0 type:0xa)	[Journal plugin.]
44	"key_short"	(id:0x0 type:0xb)	[Short key plugin.]
45	"key_large"	(id:0x1 type:0xb)	[Large key plugin.]
46	"lzo1"	(id:0x0 type:0xc)	[lzo1 compression transform plugin.]
47	"gzip1"	(id:0x1 type:0xc)	[gzip1 compression transform plugin.]
48	"none"	(id:0x0 type:0xd)	['Don't compress' compression mode plugin.]
49	"latt"	(id:0x1 type:0xd)	['Check on dynamic lattice' compression mode plugin.]
50	"ultim"	(id:0x2 type:0xd)	['Check ultimately' compression mode plugin.]
51	"force"	(id:0x3 type:0xd)	['Compress evrything' compression mode plugin.]
52	"conv"	(id:0x4 type:0xd)	['Convert to extent' compression mode plugin.]
53	"64K"	(id:0x0 type:0x10)	[64K size cluster plugin.]
54	"32K"	(id:0x1 type:0x10)	[32K size cluster plugin.]
55	"16K"	(id:0x2 type:0x10)	[16K size cluster plugin.]
56	"8K"	(id:0x3 type:0x10)	[8K size cluster plugin.]
57	"4K"	(id:0x4 type:0x10)	[4K size cluster plugin.]

- **-o, --override** *TYPE=PLUGIN, ...* – opcja umożliwia zmianę wtyczki aktualnie wykorzystywanej do obsługi danej funkcji w systemie plików. Przykładowo, zmianę algorytmu mieszającego możemy dokonać poleceniem:

```
1 [root@messy ~]# mkfs.reiser4 -y -o hash=tea_hash /dev/sda7
```

### Określanie położenia i rozmiaru dziennika

**ReiserFS 3.6** Podobnie jak w przypadku innych, dziennikowanych systemów plików, umieszczenie dziennika na innym, fizycznym urządzeniu niż sam system ReiserFS korzystnie wpływa na jego wydajność. Program *mkreiserfs* udostępnia kilka opcji umożliwiających określanie miejsca, rozmiaru maksymalnego oraz rozmiaru przeznaczonego dla pojedynczej tranzakcji. Należą do nich:

- **-j** *plik\_urządzenia* – wartością opcji jest nazwa pliku urządzenia, na którym ma zostać założony dziennik. Program *mkreiserfs* tworzy dziennik w trakcie zakładania systemu plików. Nie jest zatem konieczne jego wcześniejsze utworzenie tak, jak miało to miejsce w przypadku systemów plików ext3 czy JFS.
- **-o** *liczba\_bloków* – opcja umożliwia określenie miejsca na urządzeniu od którego ma rozpocząć się dziennik. Opcja działa jedynie wówczas, gdy dziennik znajduje się na innym urządzeniu niż system plików. W przeciwnym przypadku jest ignorowana. Jej wartością jest liczba bloków licząc od początku urządzenia, które mają zostać pominięte.

- **-s *liczba\_bloków*** – opcja służy do określania rozmiaru dziennika. Jednostką są bloki danych. Jeśli dziennik znajduje się na tym samym urządzeniu co system plików, jego rozmiar domyślny wynosi 8193 bloki. Rozmiar maksymalny nie może przekroczyć 32749 (dla systemu plików wykorzystującego bloki danych o rozmiarze 4096 B). Rozmiar minimalny, niezależnie od położenia dziennika nie może być mniejszy niż 513 bloków. Rozmiar maksymalny dla dziennika położonego na innym urządzeniu niż system plików jest limitowany jedynie rozmiarem urządzenia. Założenie dziennika o rozmiarze większym niż domyślny wymaga użycia opcji **-f**. Program *mkreiserfs* twierdzi bowiem, że duży rozmiar dziennika spowoduje wydłużenie czasu zakładania systemu plików oraz czasu jego montowania i na tym kończy swoje działanie.
- **-t *liczba\_bloków*** – pozwala określić liczbę bloków dziennika przeznaczonych na jedną transakcję. Wartość domyślna wynosi 1024. Wartość maksymalna nie może przekraczać połowy rozmiaru dziennika.

Utworzenie systemu plików ReiserFS z dziennikiem znajdującym się na innym urządzeniu wymaga zatem podczas uruchamiania komendy *mkreiserfs* użycia opcji **-j**. Jeśli rozmiar dziennika ma być różny od rozmiaru domyślnego, konieczne staje się użycie opcji **-s** oraz **-f**. Jeśli dziennik systemu plików ReiserFS znajduje się na innym urządzeniu, to konieczne jest podawanie nazwy pliku urządzenia podczas montowania oraz wykorzystywania większości programów narzędziowych.

W systemie została utworzona partycja */dev/sdc1* o rozmiarze 72261 bloków 1 kB. Skoro system plików ReiserFS używa bloków 4 kB, to ich liczba w obrębie utworzonej partycji wynosi 18064. Stąd postać komendy jest następująca:

```

1 [root@messy ~]# mkreiserfs -j /dev/sdc1 -s 18064 -df /dev/sda7
2 mkreiserfs 3.6.21 (2009 www.namesys.com)
3 .....
4 Guessing about desired format.. Kernel 2.6.27.1 is running.
5 Block 16 (0x807) contains super block. Format 3.6 with non-standard journal
6 Count of blocks on the device: 132528
7 Number of blocks consumed by mkreiserfs formatting process: 23
8 Free blocks: 132505
9 Blocksize: 4096
10 Hash function used to sort names: "r5"
11 Number of bitmaps: 5
12 Root block: 18
13 Tree height: 2
14 Objectid map size 2, max 972
15 Journal parameters:
16     Device [0x821]
17     Magic [0xa515aa0]
18     Size 18064 blocks (including 1 for journal header) (first block 0)
19     Max transaction length 1024 blocks
20     Max batch size 900 blocks
21     Max commit age 30
22 Space on this device reserved by journal: 0
23 Filesystem state 0x0
24 sb_version 2
25 inode generation number: 0

```

```

26 | UUID: 1efe33d1-8c28-42bc-ab57-e7874c067367
27 | Initializing journal - 0%....20%....40%....60%....80%....100%
28 | Syncing..ok

```

Linia 18 informuje, że rozmiar dziennika jest identyczny z wymaganym. Liczba bloków przeznaczonych dla pojedynczej transakcji pozostała niezmienną i równa domyślnej – linia 19. Umieszczenie dziennika na innym urządzeniu pozwoliło zwiększyć liczbę dostępnych bloków danych dokładnie o jego rozmiar (8193 bloki o rozmiarze 4 kB).

## Reiser4

### Montowanie systemu plików

**ReiserFS 3.6** Montowanie systemu plików ReiserFS można przeprowadzić wykorzystując standardowy, linuksowy program *mount*. Oprócz opcji ogólnych, możliwych do wykorzystania dla każdego typu systemu plików, posiada on opcje dedykowane. Mogą one zostać użyte w wierszu polecenia, po opcji **-o** lub w pliku */etc/fstab*. Do najczęściej wykorzystywanych należą:

- *conv* – użycie opcji niesie informację, że montowany system plików jest zgodny z wersją 3.5. W tak zamontowanym systemie plików, wszystkie obiekty będą tworzone w formacie wykorzystywanym w wersji 3.6 systemu plików ReiserFS. Nie będzie go można później montować jako systemu w wersji 3.5. Dodatkowo należy użyć opcji **-hush** z wartością *detect*.
- *hash* – opcja pozwala określić stosowaną w montowanym systemie plików funkcję mieszającą. Możliwe wartości opcji są identyczne jak dla opcji **-h** programu *mkreiserfs*.
- *hashed\_relocation* – opcja wymusza dostrojenie modułu alokacji bloków danych, czego zauważalnym efektem jest wzrost wydajności systemu plików.
- *jdev=nazwa\_pliku\_urzadzenia* – opcję wykorzystuje się jeśli dziennik systemu plików został założony na innym urządzeniu niż system plików. Wówczas jako wartość opcji podaje się nazwę pliku reprezentującego urządzenie zawierające dziennik.
- *no\_unhashed\_relocation* – podobnie jak w przypadku opcji *hashed\_relocation*.
- *noborder* – opcja umożliwia wyłączenie algorytmu tzw. alokacji brzegowej. Może być stosowana podczas testowania wydajności systemu plików, natomiast nie zaleca się jej użycia podczas montowania normalnie pracujących systemów plików.
- *nolog* – opcja pozwala na wyłączenie mechanizmu dziennikowania na poziomie zapisu operacji w dzienniku. Pozostałe jego elementy pozostają aktywne. Opcja stosowana jest dla podniesienia efektywności systemu plików. Należy być świadomym, że po awarii systemu plików działającego bez dziennikowania sprawdzenie jego spójności programem *reiserfsck* będzie trwało znacznie dłużej niż w przypadku kiedy pracował on z włączonym mechanizmem dziennikowania.
- *notail* – użycie opcji powoduje wyłączenie mechanizmu *tail-packing* umożliwiającego przechowywanie danych z małych plików bezpośrednio w węzłach B+ drzewa. Mechanizm umożliwia racjonalną gospodarkę blokami danych systemu plików.

- *replayonly* – użycie opcji powoduje odtworzenie wszystkich transakcji zatwierdzonych do wykonania i zapisanych w dzienniku ale nie montuje systemu plików. Jest ona wykorzystywana przez program *reiserfsck* przed rozpoczęciem sprawdzania hipotetycznie uszkodzonego systemu plików. Podczas normalnego montowania systemu plików nie powinna być stosowana.
- *resize=liczba\_bloków* – opcję stosuje się do podania rzeczywistego rozmiaru montowanego systemu plików. Informacja zapisana w bloku identyfikacyjnym jest wówczas ignorowana. Rozwiązanie to jest stosowane podczas operacji zmiany rozmiaru systemu plików komendą *resize\_reiserfs*.
- *user\_xattr* – umożliwia stosowanie rozszerzonych atrybutów użytkownika.
- *acl* – umożliwia obsługę list kontroli dostępu.

Przypomnijmy, że system plików ReiserFS jest „niezgodny” z linuksowym programem *dump*. Stąd zalecane jest wpisanie w przedostatnim polu pliku */etc/fstab* cyfry 0, co zapewni, że definowany system będzie pomijany podczas automatycznego procesu archiwizacji. Kolejna niezgodność dotyczy programu *reiserfsck*. Program ten działa w sposób interaktywny, co nie pozwala na jego wykorzystanie podczas inicjalizacji systemu, kiedy dialog z użytkownikiem nie jest możliwy. Stąd w ostatnim polu linii opisującej system plików ReiserFS również powinna pojawić się cyfra 0.

## Reiser4

### Kontrola spójności systemu plików

**ReiserFS 3.6** Dla systemu plików ReiserFS zaimplementowany został program służący weryfikacji jego spójności. Dostarczany w pakiecie *reiserfsprogs*, po skompilowaniu jest instalowany w katalogu */usr/local/sbin* w pliku o nazwie *reiserfsck*. Dla zachowania spójności można utworzyć dowiązanie symboliczne w katalogu */sbin* o nazwie *fsck.reiser*.

Uruchomienie programu *reiserfsck* bez opcji i argumentów informuje o sposobie uruchomienia oraz udostępnia krótki opis możliwych trybów pracy i sposobie działania opcji:

```
1 [root@messy scratch1]# reiserfsck
2 Usage: reiserfsck [mode] [options] device
```

Program wymaga jedynie podania nazwy pliku urządzenia. Do wyboru mamy jeden z pięciu możliwych trybów pracy:

1. **--check** – jest to domyślny tryb pracy. Program przeprowadza analizę spójności systemu plików, informuje o znalezionych nieprawidłowościach, jednak nie podejmuje żadnej próby naprawy.
2. **--fix-fixable** – w tym trybie program *reiserfsck* dokonuje naprawy jedynie tych niespójności, które w konsekwencji nie wymagają przebudowy B+ drzewa systemu plików.
3. **--rebuild-sb** – konsekwencją użycia tej opcji jest odtworzenie informacji przechowywanych w bloku identyfikacyjnym systemu plików. Praktycznie opcję tę wykorzystuje się jeśli program *mount* nie jest w stanie zamontować systemu plików i wypisuje następujący komunikat: *read\_super\_block: can't find a reiserfs file system*.

4. **--rebuild-tree** – ten tryb pracy pozwala na odbudowę B+ drzewa systemu plików, z wykorzystaniem informacji znalezionej w systemie plików. Program *reiserfsck* należy uruchomić w tym trybie, jeśli uruchomiony w trybie **--check** wypisał komunikat: „Running with **--rebuild-tree is required**”. Należy pamiętać, aby nie wykonywać kopii zapasowej systemu plików ReiserFS jeśli uprzednio nie zweryfikowano jego spójności programem *reiserfsck* uruchomionym z tą właśnie opcją oraz, aby nie wykonywać żadnych zapisów w systemie plików jeśli sprawdzanie spójności w tym trybie nie zostanie zakończone.
5. **--clean-attributes** – praca programu *reiserfsck* w tym trybie służy wyzerowaniu stosowanych w systemie ReiserFS dodatkowych atrybutów. W pewnym sensie opcja ma charakter przyszłościowy. Zakłada się, że w kolejnych wersjach systemu występowanie atrybutów rozszerzonych nie będzie konieczne, gdyż ich funkcjonalność zostanie przeniesiona do kodu zawartego w jądrze systemu operacyjnego. Aby tak zaimplementowane jądro nie miało problemów z obsługą starszych wersji systemów plików zaleca się w nich zerowanie atrybutów.

Program posiada także kilka opcji, do których należą:

- **-j plik\_urzadzenia\_z\_dziennikiem** – opcja służy wskazaniu urządzenia zawierającego dziennik systemu plików. Stosuje się ją jeśli dziennik systemu plików znajduje się na innym urządzeniu.
- **-B plik\_blokow\_uszkodzonych** – wartością opcji jest ścieżka dostępu do pliku wykorzystującego bloki uszkodzone. Podczas wykonywania programu *reiserfsck* lista bloków uszkodzonych jest tworzona od nowa. Opcję tę zaleca się stosować w trybie **--fix-fixable**. Jeśli okaże się, że urządzenie zawiera bloki uszkodzone, to konieczna jest weryfikacja spójności w trybie **--rebuild-tree**.
- **-l nazwa\_pliku\_dziennika** – opcja pozwala na wskazanie pliku, do którego zostaną zapisane informacje generowane podczas pracy programu (dziennik weryfikacji spójności).
- **-n** – opcja wymusza na programie, aby nie wypisywał żadnych informacji o znalezionych niespójnościach.
- **-z** – opcję wykorzystuje się do korekty rozmiaru plików, których rozmiar jest większy niż odległość od ostatniego znalezionej bajtu. Dzięki temu powstałe za końcem pliku wolne obszary zostaną zlikwidowane. Rozmiary plików o wartościach mniejszych niż odległość od ostatniego znalezionej bajtu mogą zostać skorygowane podczas pracy w trybie **--fix-fixable**.
- **-q** – to opcja „kosmetyczna”. Powoduje, że program nie informuje o stopniu zaawansowania prac.
- **-y** – opcja wymusza konieczność potwierdzania operacji, które program ma wykonać w celu przywrócenia spójności systemu plików. Z oczywistych względów opcja nie działa w trybie **--rebuild-tree**.
- **-f** – opcja jest wykorzystywana do wymuszenia sprawdzenia spójności systemu plików, jeśli system plików jest zaznaczony jako „czysty”.
- **-no\_journal\_available** – jest pierwszą z tzw. opcji eksperckich. Wymusza ona pracę programu jeśli dziennik systemu plików nie jest dostępny. Sytuacja taka może mieć miejsce w przypadku awarii urządzenia, na którym znajdował się zewnętrzny system plików. Jeśli system plików i jego dziennik znajdują się na tym samym urządzeniu, to opcja jest

ignorowana. Po zakończeniu pracy programu *reiserfsck* użytego z tą opcją należy wskazać położenie dziennika korzystając z programu *reiserfstune*.

- **-S** – to druga z opcji eksperckich działająca w trybie **--rebuild-tree**. Wymusza ona przeglądanie całej partycji, na której znajduje się system plików, a nie jedynie wykorzystanej przez system plików.

Tradycyjnie, dla sprawdzenia efektywności programu z rodziny fsck, uszkodzimy system plików zapisując w jego początkowym obszarze bajty o wartości losowej. Przypomnijmy, że pierwsze 64 kB to obszar, z którego system plików ReiserFS efektywnie nie korzysta. Kolejne 80 bajtów zajmuje blok identyfikacyjny. Zostanie on w całości nadpisany:

```
1 [root@messy ~]# dd if=/dev/urandom of=/dev/sda7 bs=65616 count=1
2 1+0 records in
3 1+0 records out
4 65616 bytes (66 kB) copied, 0.0958644 s, 684 kB/s
5 [root@messy ~]# mount -t reiserfs /dev/sda7 /scratch1
6 mount: wrong fs type, bad option, bad superblock on /dev/sda7,
7      missing codepage or helper program, or other error
8      In some cases useful info is found in syslog - try
9      dmesg | tail or so
```

Jak widać uszkodzony w ten sposób system plików nie może zostać zamontowany. Uruchomienie programu *reiserfsck* w trybie **--check** pozwala uzyskać następujące informacje:

```
1 [root@messy ~]# reiserfsck --check /dev/sda7
2 reiserfsck 3.6.21 (2009 www.namesys.com)
3
4 *****
5 ** If you are using the latest reiserfsprogs and it fails **
6 ** please email bug reports to reiserfs-list@namesys.com, **
7 ** providing as much information as possible -- your **
8 ** hardware, kernel, patches, settings, all reiserfsck **
9 ** messages (including version), the reiserfsck logfile, **
10 ** check the syslog file for any related information. **
11 ** If you would like advice on using this program, support **
12 ** is available for $25 at www.namesys.com/support.html. **
13 *****
14
15 Will read-only check consistency of the filesystem on /dev/sda7
16 Will put log info to 'stdout'
17
18 Do you want to run this program?[N/Yes] (note need to type Yes if you do):Yes
19
20 reiserfs_open: the reiserfs superblock cannot be found on /dev/sda7.
21 Failed to open the filesystem.
22
23 If the partition table has not been changed, and the partition is
24 valid and it really contains a reiserfs partition, then the
25 superblock is corrupted and you need to run this utility with
```

```
26 --rebuild-sb.
```

```
27
```

Zgodnie z zaleceniami program *reiserfsck* uruchamiamy w trybie umożliwiającym odzyskanie bloku identyfikacyjnego. Ponieważ dziennik analizowanego systemu plików znajduje się na innym urządzeniu, konieczne jest użycie opcji **-j**:

```
1 [root@messy ~]# reiserfsck --rebuild-sb -j /dev/sdc1 /dev/sda7
2 reiserfsck 3.6.21 (2009 www.namesys.com)
3
4 *****
5 ** If you are using the latest reiserfsprogs and it fails **
6 ** please email bug reports to reiserfs-list@namesys.com, **
7 ** providing as much information as possible -- your **
8 ** hardware, kernel, patches, settings, all reiserfsck **
9 ** messages (including version), the reiserfsck logfile, **
10 ** check the syslog file for any related information. **
11 ** If you would like advice on using this program, support **
12 ** is available for $25 at www.namesys.com/support.html. **
13 *****
14
15 Will check superblock and rebuild it if needed
16 Will put log info to 'stdout'
17
18 Do you want to run this program?[N/Yes] (note need to type Yes if you do):Yes
19
20 reiserfs_open: the reiserfs superblock cannot be found on /dev/sda7.
21
22 what the version of ReiserFS do you use[1-4]
23     (1) 3.6.x
24     (2) >=3.5.9 (introduced in the middle of 1999) (if you use linux 2.2, c
25 se this one)
26     (3) < 3.5.9 converted to new format (don't choose if unsure)
27     (4) < 3.5.9 (this is very old format, don't choose if unsure)
28     (X) exit
29 1
30
31 Enter block size [4096]:
32 4096
33
34 Did you use resizer(y/n)[n]: n
35 rebuild-sb: no uuid found, a new uuid was generated (42f03424-c59a-4d8f-b69e-cd
36 9fca183)
37
38 Enter journal offset on /dev/sdc1 in blocks [0]:
39 0
40
41 Enter journal size (including 1 block for journal header) on /dev/sdc1 in block
42 18064]:
```



```

43 rebuild-sb: generate the new journal magic (200955088)
44 Reiserfs super block in block 16 on 0x807 of format 3.6 with non-standard journal
45 Count of blocks on the device: 132528
46 Number of bitmaps: 5
47 Blocksize: 4096
48 Free blocks (count of blocks - used [journal, bitmaps, data, reserved] blocks):
49 Root block: 0
50 Tree height: 0
51 Hash function used to sort names: not set
52 Objectid map size 0, max 972
53 Journal parameters:
54     Device [0x821]
55     Magic [0xbfa54d0]
56     Size 18064 blocks (including 1 for journal header) (first block 0)
57     Max transaction length 1024 blocks
58     Max batch size 900 blocks
59     Max commit age 30
60 Blocks reserved by journal: 0
61 Fs state field: 0x1:
62     some corruptions exist.
63 sb_version: 2
64 inode generation number: 0
65 UUID: 42f03424-c59a-4d8f-b69e-cd41b9fca183
66 LABEL:
67 Set flags in SB:
68 Mount count: 1
69 Maximum mount count: 30
70 Last fsck run: Fri Nov 27 18:48:19 2009
71 Check interval in days: 180
72 Is this ok ? (y/n)[n]: y
73 The fs may still be inconsistent. Run reiserfsck --check.
74

```

Użycie programu *reiserfsck* w trybie **--check** pozwala uzyskać informację, że blok identyfikacyjny jest uszkodzony i należy go naprawić stosując tryb **--rebuild-sb**. Kolejny program uruchamiano w trybach: **--check**, **--rebuild-tree**, **--check**. Następnie system plików udało się zamontować. Jak widać proces naprawy był dość uciążliwy i długotrwały. *reiserfsck* potrafił jednak odtworzyć blok identyfikacyjny na podstawie informacji zawartej w systemie plików i jego dzienniku.

**Reiser4** Do weryfikacji spójności systemu plików Reiser4 służy program *fsck.reiser4*, znajdujący się w pakiecie *reiser4progs*, który po kompilacji zostaje zainstalowany w katalogu */usr/local/sbin*. Program wymaga podania argumentu, którym jest ścieżka dostępu do pliku urządzenia, na którym znajduje się analizowany system plików. Domyślną opcją jest **--check**. Umożliwia ona znalezienie i wypisanie informacji o występujących w systemie plików niespójnościach. Nie są jednak podejmowane żadne próby ich usunięcia. W tym trybie pracy wymagane jest, aby system plików był zamontowany w trybie tylko do odczytu.

Spśród opcji służących usuwaniu znalezionych niespójności należy wymienić:

- **--fix** – opcja umożliwia usuwanie prostych niespójności takich, które nie wymagają odtwarzania struktur systemu plików, np. zawartości bloku identyfikacyjnego czy drzewa.
- **--build-sb** – opcję wykorzystuje się jeśli uszkodzeniu uległ blok identyfikacyjny systemu plików. Pozwala ona na usunięcie występujących w nim niespójności lub odtworzenie jego zawartości z obszaru roboczego (scratcha).
- **--build-fs** – opcja pozwala na usuwanie niespójności w systemie plików występujących poza blokiem identyfikacyjnym. Jeśli to konieczne, program podejmuje próbę odtworzenia systemu plików z przestrzeni roboczej.
- **-L, --logfile nazwa\_pliku** – argumentem opcji jest ścieżka dostępu do pliku, w którym zostanie zapisana informacja o każdej znalezionej niespójności. Normalnie informacja ta jest wypisywana na wyjście diagnostyczne.
- **-n, --no-log** – użycie opcji powoduje, że żadna informacja o znalezionej niespójności nie zostanie wypisana na wyjściu diagnostycznym.
- **-a, --auto** – opcja umożliwia sprawdzenie spójności systemu plików w sposób automatyczny (bez zadawania żadnych pytań).

Program *fsck.reiser4* posiada również opcje umożliwiające wypisanie profilu systemu plików oraz manipulowanie wtyczkami. Należą do nich:

- **--print-profile** – opcja umożliwia wypisanie profilu, czyli aktualnie używanego zestawu wtyczek.
- **-l, --print-plugins** – opcja dostarcza listę wtyczek dostępnych w aktualnej bibliotece *libreiser4*.
- **-o, --override TYPE=PLUGIN, ...** – podobnie, jak w programie *mkfs.reiser4*, opcja pozwala na zmianę wtyczki (funkcji) obsługującej wybrany aspekt systemu plików.

Spośród pozostałych opcji na uwagę zasługują:

- **-y, --yes** – opcja wyłącza tryb interaktywny udzielając odpowiedź twierdzącą na każde pytanie zadane przez program.
- **-p, --preen** – opcję wykorzystuje się do usuwania prostych niespójności występujących w systemie plików.

Podobnie jak poprzednio, sprawdzimy skuteczność programu uszkadzając początek systemu plików. Nadpisując 256 kB uszkadzamy obszar zarezerwowany, główny blok identyfikacyjny, blok identyfikacyjny formatu oraz fragment mapy bitowej systemu plików:

```

1 [root@messy ~]# dd of=/dev/sda7 if=/dev/urandom bs=512 count=256
2 256+0 records in
3 256+0 records out
4 131072 bytes (131 kB) copied, 0.149285 s, 878 kB/s

```

Próba zamontowania kończy się niepowodzeniem:

### 3.4. ADMINISTRACJA WYBRANYMI SYSTEMAMI PLIKÓW W DYSTRYBUCJI REDHAT267

```
1 [root@messy ~]# mount -t reiser4 /dev/sda7 /scratch2
2 mount: wrong fs type, bad option, bad superblock on /dev/sda7,
3     missing codepage or helper program, or other error
4     In some cases useful info is found in syslog - try
5     dmesg | tail or so
```

Pierwsze uruchomienie programu *fsck.reiser4* ma na celu rozpoznanie stanu systemu plików. Stąd pojawia się jedynie argument wskazujący urządzenie, na którym system plików się znajduje:

```
1 [root@messy ~]# fsck.reiser4 /dev/sda7
2 *****
3 This is an EXPERIMENTAL version of fsck.reiser4. Read README first.
4 *****
5
6 Fscking the /dev/sda7 block device.
7 Will check the consistency of the Reiser4 SuperBlock.
8 Will check the consistency of the Reiser4 FileSystem.
9 Continue?
10 (Yes/No): y
11 ***** fsck.reiser4 started at Thu Dec 17 16:00:24 2009
12 Fatal: Wrong magic found in the master super block.
13 FSCK: backup.c: 505: repair_backup_open: Found backup does not match to the
14 on-disk filesystem metadata.
15 Fatal: Failed to open the master super block.
16 Fatal: Cannot open the FileSystem on (/dev/sda7).
17
18 1 fatal corruptions were detected in SuperBlock. Run with --build-sb option to fix them.
```

Ostatnia linia przytoczonego powyżej raportu wskazuje, iż program poprawnie rozpoznał uszkodzenie. Sugeruje również w jaki sposób można je usunąć. Zgodnie z sugestią kolejne uruchomienie ma postać:

```
1 [root@messy ~]# fsck.reiser4 --build-sb /dev/sda7
2 *****
3 This is an EXPERIMENTAL version of fsck.reiser4. Read README first.
4 *****
5
6 Fscking the /dev/sda7 block device.
7 Will build the Reiser4 SuperBlock.
8 Will check the consistency of the Reiser4 FileSystem.
9 Continue?
10 (Yes/No): y
11 ***** fsck.reiser4 started at Thu Dec 17 16:08:35 2009
12 Reiser4 fs was detected on /dev/sda7.
13 Master super block (16):
14 magic:          ReIsEr4
15 blksize:        4096
16 format:         0x0 (format40)
```

```

17  uuid:          e733281b-4f1e-4c93-8f9a-76a908d95bb1
18  label:        <none>
19
20  Format super block (17):
21  plugin:        format40
22  description:    Disk-format plugin.
23  version:        0
24  magic:          ReIsEr40FoRmAt
25  mkfs id:        0x7252fe11
26  flushes:        2315820312219539422
27  blocks:         132528
28  free blocks:    7291528313876526254
29
30  root block:     5040478673190442767
31  tail policy:    0x2 (smart)
32  next oid:       0xd9d680dee7a68aa7
33  file count:     11673420313130393815
34  tree height:    17714
35  key policy:     LARGE
36
37
38  CHECKING THE STORAGE TREE
39  Warn : Reiser4 storage tree does not exist. Filter pass skipped.
40      Read nodes 0
41      Nodes left in the tree 0
42      Leaves of them 0, Twigs of them 0
43      Invalid node pointers 1
44      Time interval: Thu Dec 17 16:08:35 2009 - Thu Dec 17 16:08:36 2009
45  CHECKING EXTENT REGIONS.
46      Read twigs 0
47      Time interval: Thu Dec 17 16:08:36 2009 - Thu Dec 17 16:08:36 2009
48  Warn : Fatal corruptions were found. Semantic pass is skipped.
49  ***** fsck.reiser4 finished at Thu Dec 17 16:08:36 2009
50  Closing fs...done
51
52  1 fatal corruptions were detected in FileSystem. Run with --build-fs option to fix them.

```

Program odtworzył zawartość bloku identyfikacyjnego (linie 20–35). Analizując system plików stwierdził uszkodzenie drzewa oraz obszaru wpisów. Stąd w ostatniej linii kolejna sugestia. Uruchomienie programu *fsck.reiser4* z opcją **–build-fs** umożliwiło usunięcie powstałych niespójności w taki sposób, że możliwe stało się zamontowanie systemu plików i odzyskanie informacji z plików, które zajmowały nieuszkodzone bloki. Na uwagę zasługuje krótki czas działania programu.

### Programy narzędziowe

**ReiserFS 3.6** W pakiecie *reiserfsprogs* oprócz omówionych programów służących zakładaniu systemu plików oraz weryfikacji jego spójności znajdują się jeszcze:

- *debugreiserfs* – jak mówi podręcznik systemowy, program w niektórych przypadkach pozwala rozwiązać problemy z systemem plików ReiserFS, przede wszystkim dostarczając

precyzyjnych informacji o jego stanie.

- *reiserfstune* – przeznaczeniem programu jest zmiana wartości niektórych atrybutów systemu plików.
- *resize\_reiserfs* – program pozwala zmieniać rozmiar niezamontowanego systemu plików.

Pierwszy z programów, *debugreiserfs*, uruchomiony bez opcji, a jedynie z argumentem będącym nazwą urządzenia, na którym znajduje się system plików wypisuje informacje z bloku identyfikacyjnego. Liczne opcje umożliwiają uzyskanie następujących informacji:

- *-j plik\_urządzenia* – opcja powoduje wypisanie zawartości dziennika znajdującego się na urządzeniu reprezentowanym przez wskazany plik. W naszym przypadku działa ona następująco:

```

1 [root@messy ~]# debugreiserfs -j /dev/sdc1 /dev/sda7
2 debugreiserfs 3.6.21 (2009 www.namesys.com)
3
4 Filesystem state: consistent
5
6 Reiserfs super block in block 16 on 0x807 of format 3.6 with non-standard journal
7 Count of blocks on the device: 132528
8 Number of bitmaps: 5
9 Blocksize: 4096
10 Free blocks (count of blocks - used [journal, bitmaps, data, reserved] blocks): 112315
11 Root block: 21
12 Filesystem is clean
13 Tree height: 4
14 .....
15 Journal header (block #18063 of /dev/sdc1):
16     j_last_flush_trans_id 19
17     j_first_unflushed_offset 31
18     j_mount_id 11
19     Device [0x821]
20     Magic [0xbfa54d0]
21     Size 18064 blocks (including 1 for journal header) (first block 0)
22     Max transaction length 1024 blocks
23     Max batch size 900 blocks
24     Max commit age 30
25 Mountid 10, transid 10, desc 0, length 2, commit 3
26 #0      1->16 2->18
27 Mountid 10, transid 11, desc 4, length 1, commit 6
28 #0      5->16
29 Mountid 10, transid 12, desc 7, length 1, commit 9
30 #0      8->16
31 Mountid 10, transid 13, desc 10, length 1, commit 12
32 #0     11->16
33 Mountid 10, transid 14, desc 13, length 1, commit 15
34 #0     14->16
35 Mountid 11, transid 15, desc 16, length 1, commit 18
36 #0     17->16

```

```

37 Mountid 11, transid 16, desc 19, length 1, commit 21
38 #0      20->16
39 Mountid 11, transid 17, desc 22, length 1, commit 24
40 #0      23->16
41 Mountid 11, transid 18, desc 25, length 1, commit 27
42 #0      26->16
43 Mountid 11, transid 19, desc 28, length 1, commit 30
44 #0      29->16

```

Opcja **-p** pozwala na umieszczenie zawartości dziennika wraz z innymi metadanymi systemu plików w archiwum.

- **-J** – użycie opcji umożliwia wypisanie jedynie nagłówka dziennika.
- **-d** – opcja wypisuje informację przechowywaną w węzłach drzewa systemu plików.
- **-D** – opcja wypisuje informację ze wszystkich wykorzystanych bloków danych systemu plików.
- **-m** – opcja wypisuje zawartość mapy bitowej.
- **-o** – opcja wypisuje zawartość mapy obiektów.
- **-B nazwa\_pliku** – użycie opcji spowoduje odszukanie w drzewie systemu plików bloków uszkodzonych, a następnie zapisze ich zawartość w pliku, którego nazwa stanowi wartość opcji.
- **-1 numer\_bloku** – wypisuje informację o zawartości bloku systemu plików o numerze, który został podany jako wartość opcji. Przykładowo:

```

1 [root@messy ~]# debugreiserfs -1 302 /dev/sda7
2 debugreiserfs 3.6.21 (2009 www.namesys.com)
3
4 302 is used in ondisk bitmap
5
6 =====
7 LEAF NODE (302) contains level=1, nr_items=3, free_space=196 rdkey (real items 3)
8 -----
9 |###|type|ilen|f/sp| loc|fmt|fsck|          key|
10 |  |  |  |e/cn|  |  |need|          |
11 -----
12 | 0|2 1484 0x1 DRCT (2), len 3208, location 888 entry count 0, fsck need 0, format new|
13 -----
14 | 1|2 1485 0x0 SD (0), len 44, location 844 entry count 65535, fsck need 0, format new|
15 (NEW SD), mode drwxr-xr-x, size 1144, nlink 2, mtime 24/2009 15:13:00 blocks 3, uid 0
16 -----
17 | 2|2 1485 0x1 DIR (3), len 552, location 292 entry count 18, fsck need 0, format old|
18 ###: Name          length      Object key          Hash          Gen number
19 0: "."              "( 1)          [2 1485]          0            1, loc 544, sta
20 1: "..              "( 2)          [2 105]           0            2, loc 536, sta
21 2: "errorhandler.h  "( 14)         [2 1518]         57643392      0, loc 520, sta

```

22	3: "resourcecached.h	"( 16)	[2 1487]	148213120	0,	loc 504,	state 4	"r
23	4: "sortmode.h	"( 10)	[2 1493]	159806720	0,	loc 488,	state 4	"r
24	5: "stdaddressbook.h	"( 16)	[2 1488]	190573824	0,	loc 472,	state 4	"r
25	6: "resource.h	"( 10)	[2 1515]	250834816	0,	loc 456,	state 4	"r
26	7: "distributionlist.h	"( 18)	[2 1510]	258655360	0,	loc 432,	state 4	"r
27	8: "geo.h	"( 5)	[2 1501]	292605440	0,	loc 424,	state 4	"r
28	9: "key.h	"( 5)	[2 1494]	303127040	0,	loc 416,	state 4	"r
29	10: "address.h	"( 9)	[2 1498]	472179072	0,	loc 400,	state 4	"r
30	11: "addresseedialog.h	"( 17)	[2 1508]	563444352	0,	loc 376,	state 4	"r
31	12: "vcardconverter.h	"( 16)	[2 1514]	647904384	0,	loc 360,	state 4	"r
32	13: "distributionlistdialog.h	"( 24)	[2 1497]	681160448	0,	loc 336,	state 4	"r
33	14: "format.h	"( 8)	[2 1504]	702410880	0,	loc 328,	state 4	"r
34	15: "addresseelist.h	"( 15)	[2 1520]	750378368	0,	loc 312,	state 4	"r
35	16: "phonenumbers.h	"( 13)	[2 1496]	787627520	0,	loc 296,	state 4	"r
36	17: "sound.h	"( 7)	[2 1513]	825736960	0,	loc 288,	state 4	"r
37	=====							

- **-p** – to właściwie jedna z dwóch użytecznych opcji programu. Umożliwia ona odczytanie wszystkich metadanych systemu plików. Mogą one następnie zostać zarchiwizowane oraz wykorzystane do odbudowy systemu plików. Dane zapisane w plikach są pomijane. Zastosowanie opcji może być następujące:

```

1 [root@messy ~]# debugreiserfs -p -j /dev/sdc1 /dev/sda7 | gzip -c > /tmp/sda7.gz
2 debugreiserfs 3.6.21 (2009 www.namesys.com)
3
4 Loading on-disk bitmap .. 20213 bits set - done
5 super block..ok
6 bitmaps..(5).. ok
7 journal (from 0 to 18063)..ok
8 Super block, bitmaps, journal - 18070 blocks - done, 15341 blocks left
9 0%....20%....40%....60%....80%....100% left 0, 3835 /sec
10 Packed 18070 blocks:
11     compressed 1
12     full blocks 18069
13         leaves with broken block head 0
14         corrupted leaves 0
15         internals 0
16         descriptors 0
17 data packed with ratio 1.00
18 [root@messy ~]# ls -l /tmp/sda7.gz
19 -rw-r--r-- 1 root root 157809 2009-11-30 17:11 /tmp/sda7.gz

```

- **-u** – opcja umożliwia odtworzenie systemu plików wykorzystując obraz utworzony przy pomocy opcji **-p**. Powstały w ten sposób system plików nie będzie identyczny z tym, którego obraz wykorzystano. Będzie on zawierał metadane oryginału, ale jego struktura będzie stworzona od podstaw. Użycie opcji może być następujące:

```

1 [root@messy ~]# gunzip -c /tmp/sda7.gz | debugreiserfs -u -j /dev/sdc2 /dev/sdb2
2 debugreiserfs 3.6.21 (2009 www.namesys.com)

```

```

3
4 There were 132528 blocks on the device
5 .....
6 .....
7 .....
8 .....Unpacked 1 leaves, 18069 full blocks
9 Temp file opened by fsck: ".bitmap" ..

```

Dla pomysłu wykonania powyższej operacji konieczne jest utworzenie partycji dla systemu plików oraz jeśli system plików, który posłużył do utworzenia obrazu posiadał dziennik umieszczony na urządzeniu zewnętrznym, również partycji dla dziennika. Wymagane jest, aby rozmiar tych partycji nie był mniejszy niż rozmiar partycji oryginalnych, gdyż w przeciwnym wypadku odtworzenie nie będzie możliwe. Dodatkowo odtwarzanie systemu z dziennikiem na urządzeniu zewnętrznym wymaga użycia opcji **-j** dla wskazania pliku urządzenia, na którym zostanie odtworzony dziennik.

- **-S** – opcję tę stosuje się z opcją **-p** dla wymuszenia odczytu metadanych wszystkich bloków systemu plików, a nie jedynie tych, które w mapie bitowej są zaznaczone jako używane.

Reasumując, najbardziej interesujące opcje programu *debugreiserfs* to **-p**, **-u** oraz **-S** umożliwiające tworzenie obrazu oraz odtwarzanie systemu plików. Pozostałe zdają się być użyteczne podczas edycji systemu plików i próby przywrócenia jego spójności „ręcznie”, jeśli program *reiserfsck* zawodzi.

Drugi z omawianych programów *reiserfstune* umożliwia zmianę wartości podstawowych atrybutów dziennika takich, jak jego rozmiar oraz maksymalna liczba bloków danych przeznaczonych dla pojedynczej transakcji. Możliwe jest także przeniesienie dziennika na inne urządzenie. W odniesieniu do atrybutów systemu plików możliwa jest zmiana jego etykiety, uniwersalnego numeru urządzenia oraz identyfikacja bloków uszkodzonych i utworzenie z nich pliku. Dodatkowo można ustalić co jaką liczbę montowań lub co ile dni będzie automatycznie przeprowadzana weryfikacja spójności systemu plików. Program wymaga podania jako argumentu nazwy pliku urządzenia, na którym znajduje się system plików. Uruchomiony w ten sposób wypisuje jedynie wartości podstawowych atrybutów. Spośród wielu opcji programu *reiserfstune* należy wskazać na następujące:

- **-j nazwa\_pliku\_urządzenia\_z\_dziennikiem** – opcja jest wykorzystywana do wskazania urządzenia, na którym znajduje się dziennik systemu plików. Jej użycie jest obowiązkowe w przypadku dziennika znajdującego się na innym urządzeniu niż system plików.
- **--no-journal-available** – użycie tej opcji wymusi pracę programu jeśli dziennik systemu plików nie jest dostępny. Jest wykorzystywana np. w przypadku awarii urządzenia, na którym znajdował się plik dziennika podczas odtwarzania go na innym urządzeniu.
- **--journal\_new\_device nazwa\_pliku\_urządzenia\_z\_dziennikiem** – opcja umożliwia wskazanie urządzenia, na którym zostanie założony dziennik dla systemu plików. Pomińnięcie wartości opcji spowoduje, że system plików będzie wykorzystywał dotychczasowy dziennik.
- **-s rozmiar\_dziennika** – opcja umożliwia ustalenie rozmiaru dziennika. Jednostką są bloki danych. Z opcji korzysta się najczęściej przy przenoszeniu dziennika na inne urządzenie.



- **-o** *odległość\_od\_początku* – wartością opcji jest liczba bloków danych po opuszczeniu których zaczyna się obszar efektywnie wykorzystywany przez dziennik systemu plików. Wartość domyślna to 0. Podobnie jak opcja **-s**, opcja ta jest wykorzystywana do określenia parametrów nowego dziennika.
- **-t** *liczba\_bloków\_danych* – opcja służy definiowaniu liczby bloków danych dziennika, które mogą zostać przeznaczone na pojedynczą transakcję. Wartość domyślna to 1024. W praktyce rozmiar ten nie może być większy od połowy rozmiaru dziennika.
- **-b** *nazwa\_pliku* – wartością opcji jest ścieżka dostępu do pliku zawierającego listę bloków uszkodzonych.
- **-B** *nazwa\_pliku* – opcja umożliwia zdefiniowanie pliku, który będzie posiadał bloki zaznaczone przez system plików jako uszkodzone. Rozwiązanie to zapewnia, że uszkodzony blok nie zostanie przydzielony żadnemu plikowi.
- **-u** *UUID* – opcja pozwala nadać nowy identyfikator uniwersalny danemu systemowi plików.
- **-l** *etykieta* – wartością opcji jest etykieta, która zostanie przypisana systemowi plików znajdującemu się na urządzeniu, którego nazwa pliku została podana jako argument wywołania programu.
- **-c** *liczba\_dni* – opcja umożliwia określenie liczby dni po upływie których spójność systemu plików zostanie automatycznie sprawdzona programem *reiserfsck* przed jego zamontowaniem. Zaleca się jednocześnie stosować opcje **-m**.
- **-m** *liczba\_montowań* – wartością opcji jest liczba montowań, po wykonaniu których spójność systemu plików zostanie automatycznie sprawdzona programem *reiserfsck*. Opcja działa efektywnie z jądrami systemu w wersji od 2.6.25, gdyż umożliwiają one obsługę licznika montowań.
- **-C** *data* – format argumentu to *YYYYMMDD[HH[MM[SS]]]*, określający datę ostatniej weryfikacji spójności systemu plików.

Program *reiserfstune* wykorzystywany jest przede wszystkim do operacji na dzienniku, a w szczególności przenoszenia go na inne urządzenie oraz zmiany jego rozmiaru i rozmiaru przeznaczonego do obsługi pojedynczej transakcji. Jest również wykorzystywany do zmiany domyślnych wartości definiujących częstotliwość automatycznej weryfikacji spójności wykonywanej przed jego zamontowaniem (liczba dni, liczba montowań).

Ostatnim programem narzędziowym pakietu jest program *resize\_reiserfs*. Służy on do zmiany rozmiaru systemu plików. Umożliwia zarówno jego zwiększanie jak i zmniejszanie. Należy zwrócić uwagę, że program nie zmienia rozmiaru partycji, na której system plików się znajduje. Stąd, zanim przystąpimy do zwiększenia rozmiaru systemu plików należy zwiększyć rozmiar partycji (jeśli system plików znajduje się na partycji logicznej LVM (podrozdział 3.6) mamy do dyspozycji programy *lvextend* lub *lvresize*. Programy z rodziny *parted* w większości przypadków nie obsługują systemu plików ReiserFS.). Jeśli rozmiar systemu plików został zmniejszony można odpowiednio zmniejszyć rozmiar partycji.

Program *resize\_reiserfs* wymaga podania nazwy pliku urządzenia, na którym znajduje się system plików. Jeśli dziennik systemu plików znajduje się na innym urządzeniu, wymagane jest użycie opcji **-j** w celu podania nazwy pliku urządzenia z dziennikiem. Takie uruchomienie umożliwia zwiększenie rozmiaru systemu plików do rozmiaru partycji lub wypisanie komunikatu informującego o tym, że system plików zajmuje całą partycję. Dla przykładu utworzono wolumen

logiczny o rozmiarze 360 MB, na którym założono system plików ReiserFS. Próba jego powiększenia dała następujący wynik:

```

1 [root@messy qtparted-0.4.5]# resize_reiserfs -j /dev/sdc1 /dev/rootvg/lv01
2 resize_reiserfs 3.6.21 (2009 www.namesys.com)
3
4 /dev/rootvg/lv01 already is of the needed size. Nothing to be done

```

Zmianę rozmiaru partycji o zadaną wartość umożliwia opcja **-s**. Jeśli rozmiar systemu plików chcemy określić bezpośrednio, to podajemy go po opcji z wyszczególnieniem jednostek: *K* – kB, *M* – MB lub *G* – GB. Jeśli zmiana rozmiaru jest podawana względem bieżącego rozmiaru, to poprzedzamy ją znakiem **-** jeśli rozmiar systemu plików ma zostać zmniejszony lub znakiem **+** jeśli rozmiar ulega zwiększeniu o zadaną liczbę jednostek.

W naszym przykładzie zwiększono rozmiar wolumenu logicznego (partycji), na którym znajduje się system plików o 156 MB. Następnie podjęto próbę zwiększenia rozmiaru systemu plików o 100 MB, co dało następujący wynik:

```

1 [root@messy qtparted-0.4.5]# resize_reiserfs -j /dev/sdc1 -s +100M /dev/rootvg/lv01
2 resize_reiserfs 3.6.21 (2009 www.namesys.com)
3
4 ReiserFS report:
5 blocksize          4096
6 block count        117760 (92160)
7 free blocks        117738 (92139)
8 bitmap block count  4 (3)
9
10 Syncing..done
11
12 resize_reiserfs: Resizing finished successfully.

```

Zwiększenie rozmiaru systemu plików do rozmiaru wolumenu logicznego miało następujący przebieg:

```

1 [root@messy qtparted-0.4.5]#resize_reiserfs -j /dev/sdc1 /dev/rootvg/lv01
2 resize_reiserfs 3.6.21 (2009 www.namesys.com)
3
4 ReiserFS report:
5 blocksize          4096
6 block count        132096 (117760)
7 free blocks        132073 (117738)
8 bitmap block count  5 (4)
9
10 Syncing..done
11
12 resize_reiserfs: Resizing finished successfully.

```

Program *resize\_reiserfs* wymaga, aby system plików, którego rozmiar jest zmieniany nie był zamontowany.

**Reiser4** Pakiet *reiser4progs*, oprócz programu służącego do zakładania systemu plików oraz programu do sprawdzania i usuwania niespójności zawiera jeszcze dwa inne. Pierwszy z nich, *debugfs.reiser4*, umożliwia śledzenie zawartości systemu plików, zarówno przechowywanych w nim danych jak i jego metadanych i struktur. Program wymaga podania jako argumentu ścieżki dostępu do pliku reprezentującego urządzenie, na którym znajduje się badany system plików. Dodatkowo, jeśli system plików jest zamontowany w trybie do zapisu i odczytu, konieczne jest użycie opcji **-f**. Takie uruchomienie powoduje, że zostaje przyjęta jako domyślna opcja **-s**, która pozwala uzyskać informację zapisaną w obu blokach identyfikacyjnych systemu plików:

```

1 [root@messy ~]# debugfs.reiser4 -f /dev/sda7
2 debugfs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by
4 reiser4progs/COPYING.
5
6 Master super block (16):
7 magic:          ReIsEr4
8 blksize:        4096
9 format:         0x0 (format40)
10 uuid:           2c41cb05-48d6-476c-96c5-24bb273e500d
11 label:          <none>
12
13 Format super block (17):
14 plugin:         format40
15 description:    Disk-format plugin.
16 version:        0
17 magic:          ReIsEr40FoRmAt
18 mkfs id:        0x63c3f07d
19 flushes:        0
20 blocks:         132528
21 free blocks:    132497
22 root block:     23
23 tail policy:    0x2 (smart)
24 next oid:       0x10000
25 file count:     1
26 tree height:    2
27 key policy:     LARGE
28
29 FS status block (21):
30 FS marked consistent

```

Spośród innych opcji programu *debugfs.reiser4* do najbardziej użytecznych należy zaliczyć:

- **-t, --print-tree** – opcja umożliwia wypisanie struktury drzewa danych systemu plików.
- **-b, --print-block *numer\_bloku*** – pozwala uzyskać zawartość bloku, którego numer został podany jako wartość opcji.
- **-j, --print-journal** – opcja pozwala uzyskać informacje o operacjach, które nie zostały jeszcze zakończone i znajdują się w dzienniku systemu plików.

- **-a, --print-alloc** – opcja pozwala uzyskać informację o danych podsystemu odpowiedzialnego za przydzielanie bloków systemu plików. Jeśli badamy zamontowany system plików, to konieczne staje się użycie opcji **-f**. W naszym przypadku sytuacja wygląda następująco:

```

1 [root@messy ~]# debugfs.reiser4 -f -a /dev/sda7
2 debugfs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by reiser4progs/COPYING.
4
5 Block allocator:
6 plugin:          alloc40
7 total blocks:    132528
8 used blocks:     31
9 free blocks:     132497
10
11          BLK          CRC          Used
12 -----
13          18 [ 0xe3c502ff ] 25
14          32736 [ 0x3ff00004 ] 2
15          65472 [ 0x3ff00004 ] 2
16          98208 [ 0x1ff80002 ] 1
17          130944 [ 0x770127ad ] 1
18
19 Block map:
20 [ 18(25) 32736(2) 65472(2) 98208(1) 130944(1) ]

```

W systemie plików za przydział bloków odpowiada wtyczka *alloc40*. Całkowita liczba bloków wynosi 132528, z czego zajętych zostało 31. Stąd liczba bloków dostępnych to 132497 (brak bloków uszkodzonych). Poniżej została wypisana mapa bloków zajętych w postaci tabelki oraz listy. Pierwsze 25 bloków zostało przydzielonych od bloku numer 18. Następne dwa od bloków 32736 oraz 65472 oraz po jednym od bloków 98208 i 130944. Suma bloków zajętych w kolejnych fragmentach wynosi 31 (suma liczb w ostatniej kolumnie tabelki lub liczb w nawiasach listy).

- **-d, --print-oid** – użycie opcji dostarcza informacji o wartościach atrybutów podsystemu przydzielającego identyfikator obiektu (identyfikator obiektu *Object IDentifier* to etykieta trwale przydzielana przechowywanemu obiektowi. Jest wykorzystywana w procesie ich indeksowania, np. w budowie drzewa obiektów. Ma najczęściej postać długiej liczby całkowitej bez znaku – jest przechowywana na 4-rech bajtach). W przypadku naszego systemu plików stan podsystemu jest następujący:

```

1 [root@messy ~]# debugfs.reiser4 -f -d /dev/sda7
2 debugfs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by reiser4progs/COPYING.
4
5 Oid allocator:
6 plugin:          oid40
7 next oid:        0x10000
8 used oids:       1

```

Podsystem obsługuje wtyczka *oid40*. Kolejną przydzieloną wartością będzie *0x10000*. Obecnie jest wykorzystywany jeden identyfikator.

- **-P, --pack-metadata** – opcja umożliwia odczyt metadanych systemu plików i wypisanie ich na standardowym wyjściu. Zaleca się przekierowanie standardowego wyjścia do pliku, gdyż dane te są w postaci binarnej. Zawartość pliku umożliwia odtworzenie identycznego, jak chodzi o strukturę wewnętrzną, systemu plików.
- **-U, --unpack-metadata** – to opcja, która pozwala odtworzyć system plików na podstawie metadanych czytanych ze standardowego wejścia, utworzonych z wykorzystaniem opcji **-P**.

Przykładowo, odtworzenie systemu plików znajdującego się na urządzeniu reprezentowanym przez plik */dev/sda7* wymaga utworzenia pliku z metadanymi, utworzenia partycji oraz odtworzenia systemu plików na podstawie zawartości pliku. Na poniższym listingu pominięto proces tworzenia partycji:

```

1 [root@messy ~]# debugfs.reiser4 -f -P /dev/sda7 > meta.sda7
2 debugfs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by
4 reiser4progs/COPYING.
5
6 [root@messy ~]# debugfs.reiser4 -U /dev/sdc2 < meta_sda7
7 debugfs.reiser4 1.0.7
8 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by reiser4progs/COPYING.
9
10 Info : The metadata were packed with the reiser4progs 1.0.7.
11 [root@messy ~]# mount -t reiser4 /dev/sdc2 /scratch3
12 [root@messy ~]# df /dev/sda7 /dev/sdc2
13 Filesystem          1K-blocks      Used Available Use% Mounted on
14 /dev/sda7             503712        9548    494164    2% /scratch2
15 /dev/sdc2             503712         124    503588    1% /scratch3

```

Istotnym jest aby partycja zawierająca klonowany system plików oraz ta, na której system ten zostanie powielony miały ten sam rozmiar. Jeśli partycja docelowa będzie mniejsza, odtwarzanie obrazu nie powiedzie się. Jeśli natomiast jej rozmiar będzie większy, to nie zostanie ona w całości wykorzystana. Po utworzeniu systemu plików należy „ręcznie” zwiększyć jego rozmiar.

- **-p, --print-profile** – komenda użyta z tą opcją dostarcza informacji o wtyczkach, które aktualnie obsługują aspekty systemu plików.
- **-l, --print-plugins** – opcja służy wypisywaniu wszystkich dostępnych w bibliotece wtyczek obsługujących system plików Reiser4.
- **-o, --override TYPE=PLUGIN, ...** – to funkcjonalność powtórzona z programów *mkfs.reiser4* oraz *fsck.reiser4*. Pozwala ona na zmianę domyślnej wtyczki, a co za tym idzie funkcji obsługującej wybrany aspekt działania systemu plików.

Ostatnim programem z pakietu *reiser4progs* jest program *measurefs.reiser4*, który dostarcza ogólnych informacji dotyczących systemu plików. Jego uruchomienie wymaga podania jako argumentu ścieżki dostępu do pliku urządzenia, na którym znajduje się badany system plików. Jeśli jest on aktualnie zamontowany, to konieczne jest użycie opcji **-f**:

```

1 [root@messy scratch2]# measurefs.reiser4 -f /dev/sda7
2 measurefs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by
4 reiser4progs/COPYING.
5
6 Tree statistics ... done
7 Packing statistics:
8   Formatted nodes:      3660.59b (89.37%)
9   Branch nodes:        1684.00b (41.11%)
10  Twig nodes:          3589.26b (87.63%)
11  Leaf nodes:          3885.24b (94.85%)
12
13 Node statistics:
14   Total nodes:         125651
15   Formatted nodes:      5689
16   Unformatted nodes:   119962
17   Branch nodes:         3
18   Twig nodes:          106
19   Leaf nodes:          125542
20
21 Item statistics:
22   Total items:         26740
23   Nodeptr items:       5688
24   Statdata items:      10070
25   Direntry items:      1494
26   Tail items:          7275
27   Extent items:        2213

```

Dostarczana informacja składa się z trzech części. Pierwsza dotyczy stopnia defragmentacji (upakowania). Kolejne to statystyki wykorzystania węzłów oraz bloków. W praktyce uzupełnia ona wiedzę o stanie systemu plików, potencjalnie dostępnych zasobach oraz ewentualnych źródłach wzrostu wydajności.

Program *measurefs.reiser4* posiada kilka opcji. Do najczęściej wykorzystywanych należą:

- **-S, --tree-stat** – to opcja domyślna. Dostarcza ogólnych statystyk drzewa organizującego dane w systemie plików, w tym dotyczące węzłów, liści, defragmentacji danych i innych.
- **-T, --tree-frag** – opcja pozwala uzyskać informacje dotyczących całkowitej defragmentacji drzewa. Raportowana wartość jest z przedziału od 0.0 (defragmentacja minimalna) do 1.0 (maksymalny stopień defragmentacji). Z dobrym przybliżeniem można przyjąć, że im wartość ta jest większa, tym prędkość sekwencyjnego odczytu danych z systemu plików staje się mniejsza, czyli czas odczytu dłuższy.
- **-D, --data-frag** – użycie opcji pozwala określić średni stopień defragmentacji plików w systemie. W praktyce oznacza to, że program określa stopień defragmentacji dla każdego pliku z osobna, a następnie dokonuje uśrednienia. Zwracana wartość jest z przedziału od 0.0 (dla właśnie utworzonego systemu plików i zapełnionego danymi) do 1.0 (maksymalna defragmentacja). Dla nowoutworzonego systemu plików, który został następnie zapełniony danymi, stopień defragmentacji będzie miał wartość bliską 0:

```

1 [root@messy scratch2]# time measurefs.reiser4 -f -D /dev/sda7
2 measurefs.reiser4 1.0.7
3 Copyright (C) 2001-2005 by Hans Reiser, licensing governed by
4 reiser4progs/COPYING.
5
6 Data fragmentation ... done
7 0.006885
8
9 real    0m2.614s
10 user    0m0.801s
11 sys     0m0.727s

```

Na uwagę zasługuje czas, w którym program narzędziowy określił stopień defragmentacji. Dla systemu plików o rozmiarze 0.5 GB przechowującego ponad 8500 plików w około 1300-stu katalogach wyniósł niewiele ponad 2.5 sekundy.

- **-F, --file-frag** *ścieżka\_dostępu\_do\_pliku* – opcja służy określeniu stopnia defragmentacji wskazanego pliku. Jest ona określana wartością z przedziału od 0.0 (najniższy stopień defragmentacji) do 1.0 (stopień najwyższy).
- **-E, --show-file** – opcja pozwala wypisać stopień defragmentacji dla każdego pliku przechowywanego w systemie plików. Należy ją stosować z opcją **-D**.

Wśród programów narzędziowych dla systemu plików Reiser4 brakuje programu, który umożliwiałby zmianę jego rozmiaru. Jest on niezbędny, jeśli system plików został założony w systemie LVM. Stanowiłby uzupełnienie mechanizmu umożliwiającego przenoszenie metadanych systemu plików na partycję o większym rozmiarze.

### 3.4.8 System plików Btrfs (B-tree file system)

B-tree file system posiada kilka nazw stanowiących wolne rozwinięcie jego krótkiej nazwy Btrfs. Są to np. „Butter FS”, „Butterfuss”, czy „Better FS”. Prace nad systemem zostały zapoczątkowane w firmie Oracle w roku 2007. Projekt jest wspierany przez wiele innych korporacji, wśród których należy wymienić: Fujitsu, FusionIO, Intel, RedHat, Strato czy SuSE. Uważany za głównego architekta tego systemu plików, Chris Mason, postawił za cel osiągnięcie optymalnego wykorzystania przestrzeni dyskowej, zarówno pod kątem miejsca jak i czasu dostępu przy zapewnieniu wysokiego stopnia niezawodności. W tym celu wykorzystano rozwiązania konstrukcyjne zaproponowane w systemie plików Reiser4 oraz dodatkowo techniki kopiowania przy zapisie, przyspieszającej przede wszystkim operacje modyfikacji B-drzew. Stąd system plików wykorzystuje m.in. alokację opartą na obszarach (*extent*), optymalizację przestrzeni dyskowej przeznaczonej dla małych plików poprzez przechowywanie ich danych w węzłach drzew, podwójne indeksowanie zawartości katalogów, mechanizm sum kontrolnych dla danych oraz metadanych, mechanizm kopii migawkowych, oraz efektywne tworzenie kopii zapasowych. System plików został wyposażony w mechanizmy paskowania (RAID 0), kopii lustrzanej (RAID 1) i ich kombinacji w postaci RAID 10 oraz możliwość sprawdzania spójności zamontowanego systemu plików. Innymi, istotnymi dla użytkowników i administratorów cechami systemu plików Btrfs są:

- Zapewnienie możliwości zmiany rozmiaru oraz przeprowadzenia defragmentacji na zamontowanym systemie plików.

- Wprowadzenie możliwości kompresji.
- Efektywny pod względem wykorzystania przestrzeni dyskowej oraz czasu dostępu mechanizm przechowywania małych plików.
- Optymalizacje uwzględniające specyfikę dysków SSD (ang. *Solid-State Drive*).
- Skalowalność gwarantująca dopasowanie do rozmiaru przestrzeni dyskowej, rozmiaru pamięci operacyjnej oraz mocy obliczeniowej procesora umożliwia jego zastosowanie zarówno w małych systemach typu smartfon, jak również w systemach serwerowych.

Ograniczenia systemu plików Btrfs nałożone na: maksymalny rozmiar pliku – 16 EB (w systemach linuksowych 8 EB), maksymalną liczbę plików –  $2^{64}$ , maksymalny rozmiar wolumenu – 16 EB, wydają się być na chwilę obecną raczej teoretyczne, niż możliwe praktycznie do osiągnięcia. W momencie powstawania niniejszej monografii system plików znajdował się w wersji testowej. Planowano zastosowanie w nim zaawansowanych metod kodowania (ang. *Transparent Encryption*) oraz kompresji danych z pominięciem powtarzających się danych (ang. *Data deduplication*).

System plików Btrfs jest dostępny na licencji GNU–GPL, stąd możliwość jego implementacji dla systemów linuksowych. Sterownik systemu plików Btrfs jest dostępny w kodzie źródłowym jądra systemu Linux od wersji 2.6.29.

### Budowa wewnętrzna

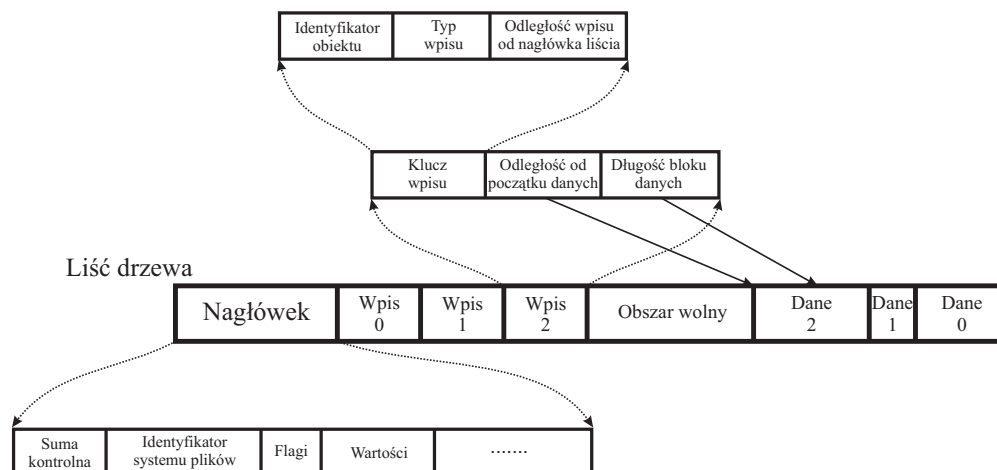
Opis budowy wewnętrznej systemu plików Btrfs wymaga zdefiniowania kilku kluczowych pojęć. Są nimi:

- **Strona** (ang. *Page*), **blok** (ang. *block*) - to ciągły obszar dysku lub pamięci operacyjnej o rozmiarze 4 kB (standardowy rozmiar dla systemów linuksowych).
- **Obszar** (ang. *Extent*) - rozumiany jako ciągły obszar przestrzeni dyskowej, zbudowany z bloków. Stąd jego rozmiar jest całkowitą wielokrotnością rozmiaru bloku.
- **Kopiowanie przy zapisie** (ang. *Copy-on-write, COW*) - jest mechanizmem wykorzystywanym przy zmianie zawartości obszaru, polegającym na zapisaniu jego zmodyfikowanej kopii w innej lokalizacji systemu plików. Dopiero po wykonaniu tej operacji oryginalny obszar jest usuwany. Operacja ta wymaga modyfikacji B–drzewa. Gwarantuje jednak wyższy poziom bezpieczeństwa danych w przypadku awarii systemu. Mechanizm ten zastosowano w systemie plików ZFS, będącym protoplastą systemu Btrfs. Tradycyjnie stosowany mechanizm uaktualniania zawartości systemu plików polega na wczytaniu modyfikowanego obszaru do pamięci, jego uaktualnieniu i zapisie w tym samym miejscu urządzenia. Tak pracujące systemy plików nazywane są *bazującymi na nadpisywaniu* (ang. *overwrite based filesystem*).

W implementacji B–drzew, w systemie plików Btrfs, istotne są trzy rodzaje danych. Są to klucze, wpisy oraz bloki nagłówkowe. Klucz opisuje lokalizację obiektu w systemie plików. Jego podstawowymi elementami są: identyfikator obiektu, typ obiektu oraz odległość początku przechowywanych danych od początku klucza. Wszystkie wartości klucza mają stałą długość, więc nie ma konieczności przechowywania informacji o długości danych. Wpis posiada również stałą długość. Wskazuje na dane, które są zazwyczaj różnej długości. Stąd, oprócz klucza, posiada również informacje o odległości od początku bloku danych do początku wskazywanego bloku danych oraz informacje o długości danego bloku danych. Bloki nagłówkowe posiadają stałą długość i są wykorzystywane do przechowywania sum kontrolnych, identyfikatorów, numeru poziomu oraz flag.



Wykorzystanie opisanych struktur danych przedstawiono na rysunku 3.20. Wewnętrzne węzły każdego drzewa przechowują informacje o kluczu oraz wskaźnikach do kolejnych węzłów. W liściach drzew przechowywane są wpisy oraz adresowane przez nie bloki danych. Wpisy przechowywane są jako tablica struktur zaczynająca się za blokiem nagłówka. Za tablicą wpisów znajduje się tablica bloków danych. Bloki danych przechowywane są w odwrotnej kolejności niż wskazujące na nie wpisy.



Rysunek 3.20: Budowa liścia B-drzewa w systemie plików Btrfs.

Każdy obiekt systemu plików, wskazywany przez wpis, posiada niepowtarzalny, 64-ro bitowy identyfikator generowany podczas tworzenia tegoż obiektu. Najstarsze bity identyfikatora to bity klucza. Podejście to pozwala na łatwe grupowanie wszystkich wpisów danego systemu plików w struktury B-drzewa. Pole typu opisuje rodzaj danych, do których odnosi się dany wpis. W kluczu przechowywana jest również informacja o miejscu, w którym zaczynają się dane przechowywane we wpisie.

Opisane struktury danych organizują dostęp do danych i metadanych składających się na system plików. Wśród nich najistotniejszymi są i-węzły opisujące pliki oraz obszary przechowujące zapisane w plikach dane. I-węzły są przechowywane we wpisach, których fragment klucza opisujący odległość wynosi zero, zaś typ to jeden. Wpisy i-węzłów posiadają zawsze najmniejszą wartość klucza dla danego obiektu i przechowują jedynie podstawowe metadane opisujące pliki lub katalogi. Ze względu na mały rozmiar wpisu i-węzła, przechowywane są w nim jedynie podstawowe wartości metadanych. Pozostałe, jak na przykład listy kontroli dostępu, przechowywane są we wpisach specjalnego typu.

Dane plików o rozmiarze mniejszym od rozmiaru bloku danych liścia są przechowywane bezpośrednio w liściach B-drzewa, z wykorzystaniem wpisów definiujących obszary. Wpis taki zawiera informację o miejscu, w którym rozpoczynają się dane oraz o ich rozmiarze. Dane plików dużych przechowywane są w obszarach, bez dodatkowych nagłówków czy formatowania. Wpis w węźle definiujący obszar składa się z numeru logicznego (w obrębie systemu plików) obszaru oraz pary: numer początkowego bloku danych na dysku (fizycznego), należącego do obszaru oraz liczby bloków dyskowych tworzących obszar. We wpisie przechowywana jest także informacja o numerze fizycznego bloku danych, w którym jest on zapisany oraz liczbie fizycznych, adresowanych przez niego bloków danych. Przechowywanie tych informacji podnosi efektywność operacji modyfikacji danych przechowywanych w środkowych blokach obszaru, gdyż pozwala wyznaczać bezpośrednio

numery bloków danych, których dane mają zostać zmienione.

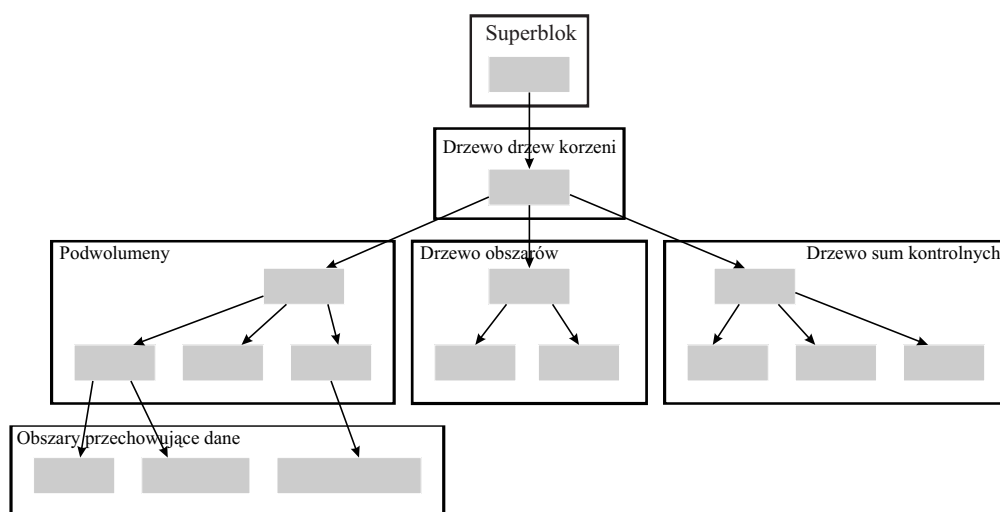
Wpisy opisujące katalogi zawierają tablice elementów składających się z nazwy katalogu (napis) oraz 64-ro bitowego identyfikatora. Dodatkowo posiadają dwa indeksy zaimplementowane jako tablice (stąd podwójne indeksowanie). Pierwsza z nich jest wykorzystywana do czytania zawartości katalogów i zawiera rekordy zbudowane z dwóch elementów: klucza wpisu definiującego katalog oraz skrótu nazwy pliku. Elementy drugiej to również rekordy zawierające dwa elementy: klucz wpisu definiującego katalog oraz numer i-węzła opisującego katalog. Ma ona przyspieszać operacje prowadzone na zawartości katalogu, w szczególności kopiowanie, przenoszenie i usuwanie plików. Jest również wykorzystywana podczas operacji sprawdzania spójności systemu plików.

**Struktura logiczna** Struktura logiczna systemu plików Btrfs to las drzew. Została ona przedstawiona na rysunku 3.21. Blok opisu partycji (superblok) jest umieszczony w tym samym miejscu partycji i stanowi punkt wejścia do systemu plików. Posiada on między innymi wskaźnik na tzw. drzewo drzew korzeni (ang. *tree of tree roots*). To z kolei zawiera wskaźniki na pozostałe B-drzewa tworzące system plików. Są nimi:

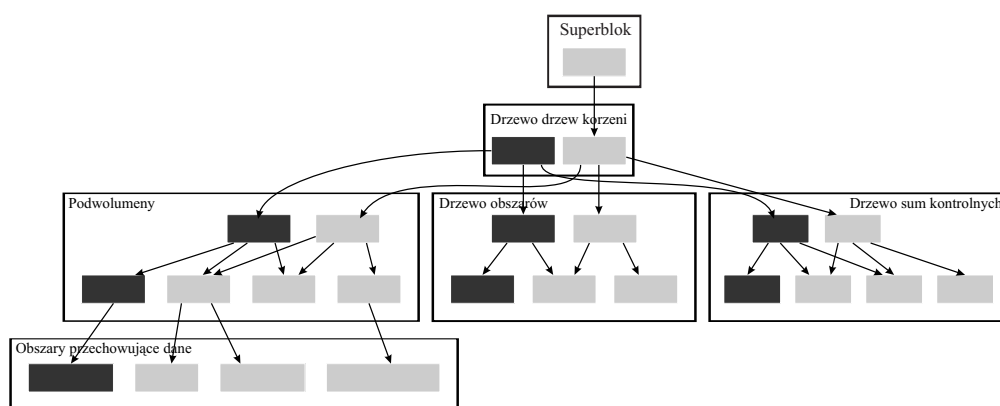
- **Drzewo podwolumenów** (ang. *Sub\_volumes*) - służy do przechowywania plików i katalogów dostępnych dla użytkowników. Każdy podwolumen jest zaimplementowany jako oddzielne B-drzewo. Dla każdego podwolumenu może zostać wykonana kopia migawkowa, jak również możliwe jest jego sklonowanie. W wyniku tych operacji zostają utworzone dodatkowe B-drzewa. Dostęp do korzeni drzew podwolumenów jest możliwy dzięki wskaźnikom przechowywanym w drzewie drzew korzeni.
- **Drzewo alokacji obszarów** (ang. *Extent allocation tree*) - służy do efektywnego zarządzania obszarami, w szczególności mapą wolnych obszarów. Wszystkie wskazania wsteczne do obszarów są pamiętane w spisach obszarów. Podejście to pozwala na przeniesienie obszaru, jeśli jest to konieczne, lub odtworzenie z uszkodzonego bloku dyskowego.
- **Drzewo sum kontrolnych** (ang. *Checksum tree*) - drzewo to przechowuje wartości sum kontrolnych dla każdego zajętego obszaru. Każda wartość składa się z listy sum kontrolnych, wyznaczonych dla każdej strony w danym obszarze.
- **Drzewa fragmentów i urządzeń** (ang. *Chunk and device trees*) - stanowi warstwę logiczną w mechanizmie zarządzania urządzeniami fizycznymi. Pozwala na tworzenie urządzeń typu RAID.
- **Drzewo realokacji** (ang. *Reloc tree*) - jest wykorzystywane podczas operacji modyfikujących wewnętrzna strukturę systemu plików, jak np. przenoszenia obszarów wykonywanej w celu zmniejszenia defragmentacji.

**Modyfikacja przechowywanej informacji** Modyfikacja plików i katalogów użytkowników pociąga za sobą zmiany zawartości stron i obszarów. Konsekwencją jest ciąg modyfikacji propagujących się w drzewie podwolumenów do jego korzenia. Modyfikacje mogą dotyczyć również alokacji obszarów, wartości liczników odwołań oraz wskaźników wstecznych. Te zmiany propagują się w drzewie obszarów. Z kolei zmiana wartości danych lub metadanych pociąga za sobą konieczność wyznaczenia nowych wartości sum kontrolnych, co może wymagać zmian w drzewie sum kontrolnych. Wszystkie te zmiany skutkują z kolei utworzeniem nowego korzenia w drzewie drzew korzeni. Przykładową modyfikację przedstawiono na rysunku 3.22.

Informacje o wszystkich modyfikacjach systemu plików są akumulowane w pamięci operacyjnej. Po upływie określonego czasu (wartość domyślna to 30 sekund) lub jeśli zmodyfikowana



Rysunek 3.21: Budowa wewnętrzna systemu plików Btrfs to las drzew organizujących fragmenty logiczne systemu plików.



Rysunek 3.22: Zmiana zawartości obszaru pociąga za sobą konieczność uaktualnienia zawartości drzewa obszarów, drzewa podwolumenów, drzewa drzew korzeni oraz drzewa sum kontrolnych. Kółem czarnym zaznaczono modyfikowane węzły drzew.

została odpowiednia liczba stron, zmiany zapisywane są na dysku, w nowym położeniu, tworząc tzw. znacznik (ang. *checkpoint*). Po zapisaniu znacznika zawartość superbloku jest modyfikowana tak, aby zawierała informacje o aktualnym stanie systemu plików oraz ostatnim znaczniku. Operacja modyfikacji superbloku jest jedyną, wykonywaną bez użycia techniki kopiowania przy zapisie (COW). Programy służące przywracaniu spójności systemu plików, po jego awarii, w pierwszej kolejności dokonują odczytu superbloku, a następnie wskaźników do ostatniego znacznika, który został zapisany na dysku. Po odczytaniu zawartości punktu kontrolnego, wszystkie nieaktualnione na dysku strony pamięci podręcznej zostają zaznaczone. System nie może wykonać żadnych zmian ich zawartości. Następnie, dla zaznaczonych stron przeprowadzana jest operacja zapisu z wykorzystaniem techniki COW. Taki sposób postępowania zabezpiecza zawar-

Plik	Zajmowany obszar
dane.txt	200–232 kB
dane.bak	200–210 kB, 512–520kB, 220–232 kB

Tablica 3.2: Przykład alokacji obszarów systemu plików dla przypadku ich współdzielenia przez dane należące do różnych plików.

tość punktu kontrolnego przez modyfikacją.

Dla drzewa podwolumenów może być wykonywana kopia migawkowa, jak również może ono zostać sklonowane. Wykonanie tych operacji wymaga działań na wskaźnikach znajdujących się strukturach drzewa, będących w tym przypadku zależnymi od położenia podwolumenu. Wszystkie pozostałe drzewa posiadają wskaźniki odnoszące się do urządzenia fizycznego oraz nie jest wykonywana dla nich operacja tworzenia kopii migawkowej. Stąd operacja przeliczania wskaźników nie jest konieczna do wykonywania.

Jak wspomniano, zmiana zawartości systemu plików pociąga za sobą konieczność zmiany wielu struktur opisujących system plików. Przykładowo zapisanie do pliku 4 kB danych, wymaga zmian w: i-węźle opisującym plik, blokach danych należących do pliku, wyznaczenia sum kontrolnych oraz referencji wstecznych. Wszystkie one powodują zmiany w stosownych drzewach. Stąd łatwo stwierdzić, że operacje zmiany zawartości przypadkowych obszarów losowo wybranych plików są najbardziej obciążające system plików. Statystycznie takie zmiany są niezwykle rzadkie. Dominują operacje o charakterze lokalnym. W szczególności są to zmiany dużej ilości danych w obrębie jednego pliku lub zmiany w plikach znajdujących się w jednym katalogu. Niemniej jednak, dla zapewnienia odpowiedniej wydajności systemu plików jego twórcy postanowili przeciwdziałać skrajnie niekorzystnym przypadkom. Przede wszystkim struktura drzew jest organizowana tak, aby wszelkie operacje na pliku skutkowały modyfikacją jedynie w obrębie jednej ścieżki w drzewie. Stąd wszelkie złożone operacje są dzielone na odpowiednie fragmenty, co dodatkowo ogranicza rozmiar punktu kontrolnego. Dodatkowo wprowadzona operacja rezerwacji superbloku zapewnia zapis punktu kontrolnego na dysku.

Technologia COW ma, jak każda, swoje zalety i wady. Niewątpliwie jest efektywnym mechanizmem gwarantującym atomowość operacji, a co za tym idzie spójność systemu plików. Jednak kluczem do wydajności systemu plików jest możliwość wykorzystywania dużych obszarów, zajmujących ciągle obszary przestrzeni dyskowej. Operacje zmiany zawartości przypadkowych obszarów plików prowadzą, przy wykorzystaniu tej technologii, do dużego rozdrobnienia obszarów i ich rozproszenia. Stąd konieczność posiadania dla systemu plików Btrfs efektywnego programu do jego defragmentacji.

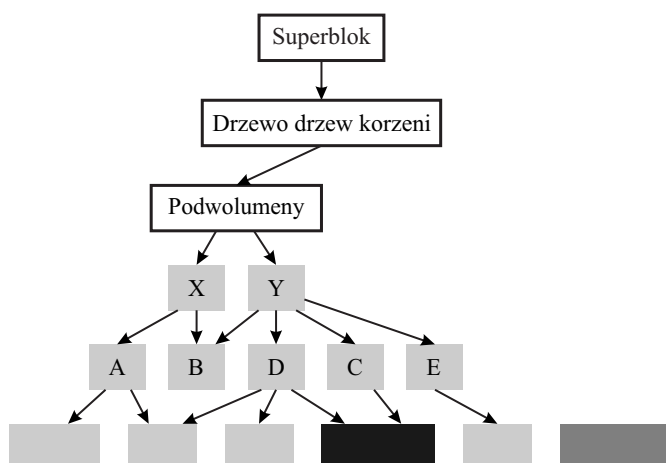
**Drzewo alokacji obszarów** zawiera opis położenia obszarów na dysku, z których każdy jest ciągłym obszarem przestrzeni adresowej. W drzewie może istnieć wiele wskazań do tego samego obszaru, ale każdy może wskazywać na jego inny fragment. Podejście to umożliwia współdzielenie fragmentów obszarów, przechowujących dane, należące do różnych plików. Załóżmy, że plik *dane.txt* zajmuje obszar od 200 do 232 kB systemu plików. Plik ten kopiujemy do pliku o nazwie *dane.bak*. Następnie zmieniamy zawartość pliku *dane.bak* w zakresie od 211 do 218 kB. Dla wykonania tej operacji konieczne jest znalezienie w systemie plików obszaru o rozmiarze 8 kB. Załóżmy, że obszar ten znaleziono we fragmencie od 512 do 520 kB systemu plików. Stąd informacja o wykorzystaniu obszarów przez dane należące do interesujących nas plików, przechowywana w drzewie alokacji obszarów będzie taka, jak przedstawiono w tabeli 3.4.8:

Należy zwrócić uwagę, iż w przedstawionym przykładzie obszar systemu plików przechowujący dane zapisane w plikach, z zakresu od 200 kB do 232 kB jest adresowany przez cztery wskaźniki. Pierwszy dotyczy pliku *dane.txt* i adresuje go w całości, wskazując jednocześnie gdzie

w systemie plików znajduje się plik *dane.bak*. Drugi wsłazuje na obszar, w którym znajdują się zmodyfikowane dane systemu plików *dane.bak*. Trzeci to wsłaznik do obszaru przechowującego ostatni, niezmodyfikowany fragment danych obu plików.

**Defragmentacja** Jak wielokrotnie wspomniano, jednym z najistotniejszych czynników wpływających na wydajność systemu plików, jest takie położenie bloków danych, aby stanowiły one ciągły obszar dysku. W pewnym zakresie problem rozwiązano wprowadzając obszary. Niestety podczas normalnej pracy systemu plików, usuwanie i tworzenie plików prowadzi do sytuacji, w której dane przechowywane w plikach znajdują się w obszarach rozproszonych po całej powierzchni dysku. Operacje defragmentacji polega na znalezieniu dla każdego pliku obszarów zajmujących ciągły obszar dysku. Wymaga ona jednak przeniesienia danych pomiędzy obszarami, co w przypadku systemu plików opartego o B-drzewa jest operacją czasochłonną. Poniżej opisano przebieg operacji przeniesienia danych między obszarami z uwzględnieniem modyfikacji B-drzew systemu plików.

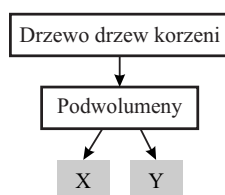
Na rysunku 3.23 przedstawiono system plików, dla którego algorytm defragmentacji znalazł obszar przechowujący dane, który nie zajmuje ciągłej przestrzeni dysku z pozostałymi obszarami. Został on zaznaczony kolorem czarnym. Dla przechowywanych przez niego danych znaleziono obszar, który zaznaczono kolorem szarym. Teraz należy przenieść dane oraz zmodyfikować odpowiednie struktury systemu plików.



Rysunek 3.23: Przykładowa zawartość systemu plików przed dokonaniem operacji defragmentacji. Kolorem czarnym zaznaczono obszar, z którego dane mają zostać przeniesione. Obszarem docelowym jest obszar zaznaczony kolorem szarym, gdyż zajmuje on ciągły obszar dysku z pozostałymi obszarami.

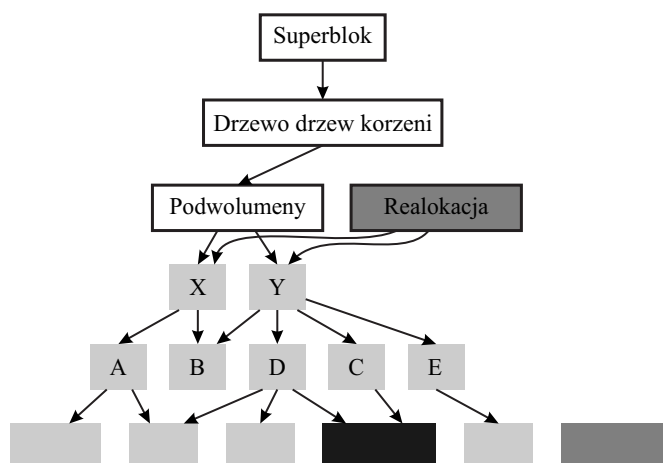
Dla przyspieszenia procesu przeglądania drzewa, w pierwszej kolejności przeglądane są referencje wsteczne dla znalezienia wszystkich węzłów położonych na wyższym poziomie drzewa, które posiadają wsłazniki pokazujące na dany fragment bezpośrednio lub pośrednio. W tym przypadku, na usuwany obszar danych, zaznaczony na rysunku 3.23 kolorem czarnym, wskazania pochodzą z węzłów oznaczonych literami *D* oraz *C*. Na te węzły wskazuje węzeł oznaczony literą *Y*. Węzeł ten wskazuje również na węzeł oznaczony literą *B*, wskazywany przez węzeł *X*. Stąd istotnymi dla przebudowy drzew są: drzewo drzew korzeni, drzewo podwolumenów oraz

obszary  $X$  oraz  $Y$  z drzewa obszarów. Na tej podstawie powstaje struktura przedstawiona na rysunku 3.24.



Rysunek 3.24: Pierwszy krok modyfikacji struktur systemu plików wynikający z konieczności przeniesienia danych pomiędzy obszarami to znalezienie węzłów drzew wskazujących na obszary przechowywane dane, które mają zostać przeniesione.

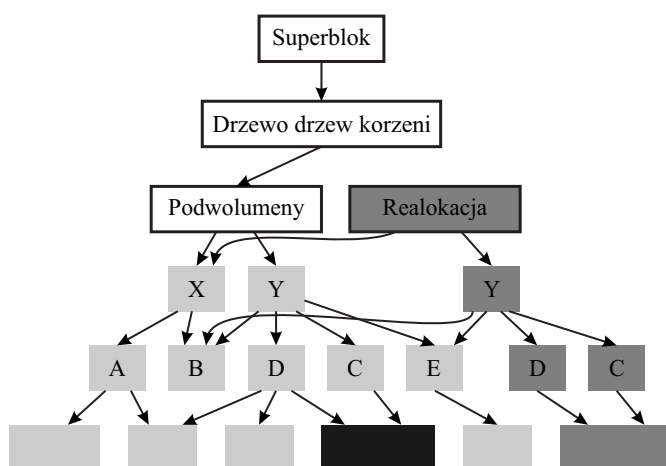
Następnie tworzona jest kopia drzewa podwolumenów. Operacja ta powoduje, iż zachowany zostaje stan drzewa podwolumenów z bieżącej chwili czasowej, zaś do kopii mogą być wprowadzane modyfikacje wynikających z defragmentacji oraz ze zmian wprowadzonych przez użytkowników w działającym systemie plików. (rysunek 3.25).



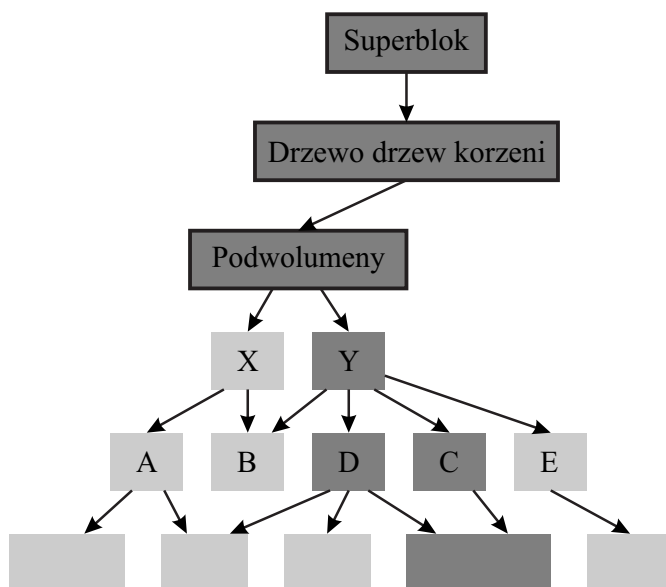
Rysunek 3.25: Organizacja systemu plików po wykonaniu kopii drzewa podwolumenów. Kopia wskazuje na węzły drzewa obszarów, których zawartość zostanie zmodyfikowana w wyniku przenoszenia danych między obszarami.

W kolejnym kroku analizowane są wskazania od kopii drzewa podwolumenów (drzewa realokacji) do węzłów drzewa obszarów w celu znalezienia tych węzłów, których zawartość zostanie zmodyfikowana w wyniku przeniesienia danych między obszarami. Jak wynika z rysunku 3.26 są to węzły oznaczone literami  $Y$ ,  $C$  oraz  $D$ . Zostają one skopiowane, a następnie zmodyfikowane tak, aby wskazywać na stosowny obszar danych. Następnie zostają skopiowane dane między obszarami. Taka kolejność operacji wynika z działania technologii kopiowania przy zapisie.

Rysunek 3.27 pokazuje stan systemu plików do dokonania przeniesienia danych między obszarami. Kolorem szarym zaznaczono zmodyfikowane węzły drzewa obszarów oraz zmodyfikowane w wyniku tej operacji drzewa: podwolumenów oraz drzew korzeni. Modyfikacji uległ również superblok, gdyż zmieniły się mapy zajętości obszarów. Jak widać proces defragmentacji w systemach plików wykorzystujących B-drzewa oraz kopiowanie przy zapisie jest złożony.



Rysunek 3.26: Znalezienie węzłów, których zawartość zostanie zmodyfikowana podczas przenoszenia danych między obszarami. Do samej modyfikacji węzłów wykorzystywana jest technologia COW, stąd pojawienie się kopii odpowiednich węzłów.



Rysunek 3.27: .

**System plików Btrfs na wielu urządzeniach fizycznych** W systemach linuxowych do zarządzania urządzeniami dyskowymi wykorzystywany jest tzw. *device-mapper* (DM). Dosłowne tłumaczenie powinno brzmieć *mapowacz urządzeń*, co mimo dziwnego brzmienia oddaje zasadę działania. Otóż został on zaimplementowany w jądrze systemu operacyjnego, w jego dolnych warstwach, a jego zadaniem jest zorganizowanie przestrzeni kilku fizycznych dysków w zadany sposób, stworzenie w ten sposób ciągłej, blokowej, wirtualnej przestrzeni adresowej i „przekazanie” jej do wyższych warstw jądra. Na tak utworzonym urządzeniu blokowym możliwe jest

utorzenie dowolnego systemu plików. Podstawowymi systemami tego rodzaju są: *Logical Volume Manager* (LVM), omówiony w podrozdziale 3.6 oraz *multiple device administrator* (mdadm) przedstawiony w punkcie 3.7.2. LVM służy do łączenia przestrzeni adresowych pochodzących z różnych urządzeń fizycznych w ciągłą przestrzeń adresową oraz posiada mechanizmy pozwalające na elastyczne zarządzanie nią. mdadm został wyposażony w mechanizmy pozwalające na tworzenie i administrowanie programowymi macierzami dyskowymi (podrozdział 3.7). Oba systemy są jednak zaimplementowane poza systemem plików. Stąd, głównie dla ułatwienia administrowania oraz podniesienia poziomu niezawodności systemu komputerowego, twórcy systemu plików Btrfs, wzorując się na rozwiązaniach zaproponowanych w systemie ZFS, zaimplementowali mechanizmy kopii lustrzanej, paskowania oraz RAID na poziomach 5 i 6.

### Tworzenie systemu plików Btrfs

Tworzenie systemu plików Btrfs jest możliwe na dwa sposoby. Pierwszy polega na przekonwertowaniu istniejącego systemu plików ext3 lub ext4 programem *btrfs-convert*. Przed konwersją należy dokonać sprawdzenia spójności konwertowanego systemu plików poleceniem *fsck.ext4* z opcją *-f*. Drugi sposób, to wykorzystanie programu *mkfs.btrfs* (lub *mkfs* z argumentem opcji *-t btrfs*) dla utworzenia nowego systemu plików.

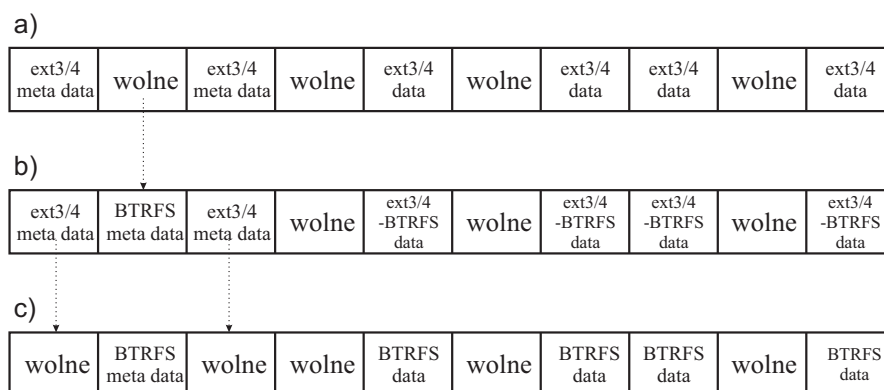
W stosunku do innych systemów plików, Btrfs posiada niewiele metadanych, które dodatkowo zapisane są w ściśle określonych miejscach systemu plików. Dodatkowo mechanizm kopiowania przy zapisie pozwala na zachowanie niemodyfikowanej kopii systemu plików z którego dokonywana była konwersja. Możliwe jest zatem wycofanie się z przeprowadzonej konwersji do oryginalnego systemu plików bez uwzględniania zmian wprowadzonych w utworzonym systemie plików Btrfs. Konwersja przebiega z wykorzystaniem funkcji zawartych w bibliotece *libe2fs*. Polega na odczytaniu metadanych konwertowanego systemu plików ext3/ext4 i wykorzystaniu jego wolnych bloków do zapisania metadanych systemu plików Btrfs. Sama konwersja przebiega w następujących krokach:

- Wykonanie kopii pierwszego MB urządzenia, na którym znajduje się konwertowany system plików.
- Wykonanie kopii katalogów i i-węzłów istniejącego systemu plików dla utworzenia systemu plików Btrfs.
- Stworzenie kopii tablic alokacji bloków danych zapisanych w i-węzłach konwertowanego systemu plików tak, aby tworzony system miał dostęp do zapisanych danych.

Przeprowadzona w ten sposób konwersja działa na metadanych, co powoduje, iż nowopowstały system plików Btrfs pracuje na blokach danych konwertowanego systemu plików. Utworzenie kopii pierwszego MB urządzenia umożliwia zapisywanie metadanych systemu plików Btrfs w tym obszarze. Stąd możliwość łatwego powrotu do konwertowanego systemu plików. Dodatkowo podczas konwersji tworzona jest kopia migawkowa, zawierająca wskazania na bloki danych wykorzystywane przez konwertowany system plików. Jej usunięcie umożliwia odzyskanie części bloków danych w systemie plików oraz powoduje, że konwersja ma charakter trwały.

Kolejne etapy konwersji zobrazowano schematycznie na rysunku 3.28. W części a) przedstawiono organizację konwertowanego systemu plików ext3/ext4. Rysunek b) przedstawia organizację systemu plików po dokonaniu konwersji. Działa system plików Btrfs. Dane przechowywane w blokach danych są współdzielone między obydwoma systemami plików. Zachowana została kopia migawkowa konwertowanego systemu plików. Organizację systemu plików po usunięciu kopii migawkowej przedstawia fragment c) rysunku.





Rysunek 3.28: Etapy tworzenia systemu plików Btrfs na drodze konwersji z systemu plików ext3/4.

Na zamieszczonym poniżej listingu przedstawiono przebieg procesu utworzenia systemu plików ext4 na partycji o rozmiarze 7GB (usunięto komunikaty programu `mkfs.ext4`) oraz jego konwersji do systemu plików Btrfs. Należy pamiętać, aby konwertowany system plików nie był zamontowany oraz aby przed konwersją przeprowadzić weryfikację jego spójności programem `fsck.ext4`. Pomiary czasu wykonania poszczególnych operacji przeprowadzono poleceniem `time`.

```

1  [root@messy ~]# time mkfs.ext4 /dev/sdc2
2  .....
3  real    0m12.295s
4  user    0m0.410s
5  sys     0m4.201s
6
7  [root@messy ~]# time fsck.ext4 -f /dev/sdc2
8  .....
9
10 [root@messy ~]# btrfs-convert /dev/sdc2
11 creating btrfs metadata.
12 creating ext2fs image file.
13 cleaning up system chunk.
14 conversion complete.
15
16 real    0m9.341s
17 user    0m0.324s
18 sys     0m3.418s

```

Łączny, przybliżony czas tworzenia w ten sposób systemu plików Btrfs to ok. 22 s. Proces konwersji do systemu plików Btrfs utworzył system plików z jednym podwolumenem. W chwili obecnej możemy zamontować „przejściowy” system plików Btrfs oraz kopie migawkową systemu plików ext4. W momencie montowania zawartość obu systemów plików jest identyczna. Polecenia są następujące:

```

1  [root@messy ~]# mount -t btrfs /dev/sdc2 /mnt/btrfs
2  [root@messy ~]# mount -t btrfs -o subvol=ext2_saved /dev/sdc2 /mnt/ext2_saved

```

```
3 [root@messy ~]# mount -t ext4 -o loop,ro /mnt/ext2_saved/image /mnt/ext4
```

Przechowywanie kopii migawkowej umożliwia powrót do systemu plików ext3/ext4 i odtworzenie go w stanie takim, w jakim był przed konwersją. Przed wykonaniem tej operacji system plików należy odmontować, a następnie użyć programu *btrfs-convert* opcją *-r*, co przedstawiono poniżej:

```
1
```

### Zmiana wartości atrybutów systemu plików Btrfs

Do zmiany wartości atrybutów służy program narzędziowy *btrfstune*. Indeks *btrfs-tune*. W porównaniu do programów tego typu dla innych systemów plików jego funkcjonalność jest bardzo skromna. Podstawowe opcje programu to:

- **-S** *wartość\_przesunięcia* –
- **-r** – włącza obsługę wskaźników do rozszerzonych i-węzłów.
- **-x** – włącza obsługę wskaźników do meadanych.

### Montowanie systemu plików Btrfs

#### Wybrane programy narzędziowe

System plików Btrfs udostępnia relatywnie dużo programów narzędziowych. Oprócz podstawowych, służących weryfikacji spójności systemu plików oraz jego strojeniu, pojawiło się wiele innych. Konieczność ich istnienia jest podyktowana budową wewnętrzną oraz wynikającymi z niej mechanizmami działania. Do podstawowych programów narzędziowych należą:

- *btrfs* – to podstawowy program narzędziowy systemu plików Btrfs, służący do zarządzania plikami i katalogami przechowywanymi w systemie. Umożliwia również tworzenie i usuwanie kopii migawkowej podwolumenu, przeprowadzanie defragmentacji systemu plików, zmianę jego rozmiaru, skanowanie zawartości, synchronizację oraz dodawanie i usuwanie urządzeń do puli wykorzystywanej przez system plików.
- *btrfsctl* – jest programem, który służy do zarządzania systemem plików. Jego funkcjonalność została zawarta w programie *btrfs* i aktualnie nie jest rozwijany.
- *btrfs-debug-tree* – program umożliwia zapisanie zawartości całego drzewa znajdującego się na urządzeniu reprezentowanym przez plik będący argumentem uruchomienia polecenia. Informacja ta może być przydatna do „ręcznego” śledzenia i usuwania niespójności systemu plików. Opcje umożliwiają wybór informacji dotyczących wybranych elementów drzewa jak np. obszarów, katalogu głównego, korzeni, czy konkretnego bloku.
- *btrfs-find-root* – umożliwia znajdowanie numeru bloku zawierającego korzeń drzewa. Dostępne opcje pozwalają na filtrowanie po numerze identyfikacyjnym obiektu, numerze poziomu oraz po numerze transakcyjnym.
- *btrfs-image* – program pozwala na tworzenie obrazu systemu plików. Tworząc obraz dane są zerowane, natomiast metadane zachowane. Istnieje możliwość kompresji obrazu na różnym stopniu, jak również tworzenia kopii wybranych bloków. Ta ostatnia możliwość jest

wykorzystywana przy tworzeniu obrazu uszkodzonego drzewa obszarów dla odzyskania jego zawartości. Program umożliwia także odtwarzanie systemu plików z utworzonego uprzednio obrazu.

- *btrfs-map-logical* –
- *btrfs-restore* –
- *btrfs-show* – przegląda katalog */dev* w celu znalezienia pliku urządzenia, na którym istnieje system plików Btrfs. W przypadku znalezienia wypisuje jego krótką charakterystykę w postaci etykiety, numeru UUID, rozmiaru, wykorzystanej i dostępnej, itd.
- *btrfstune* –
- *btrfs-vol* –
- *btrfs-zero-log* –

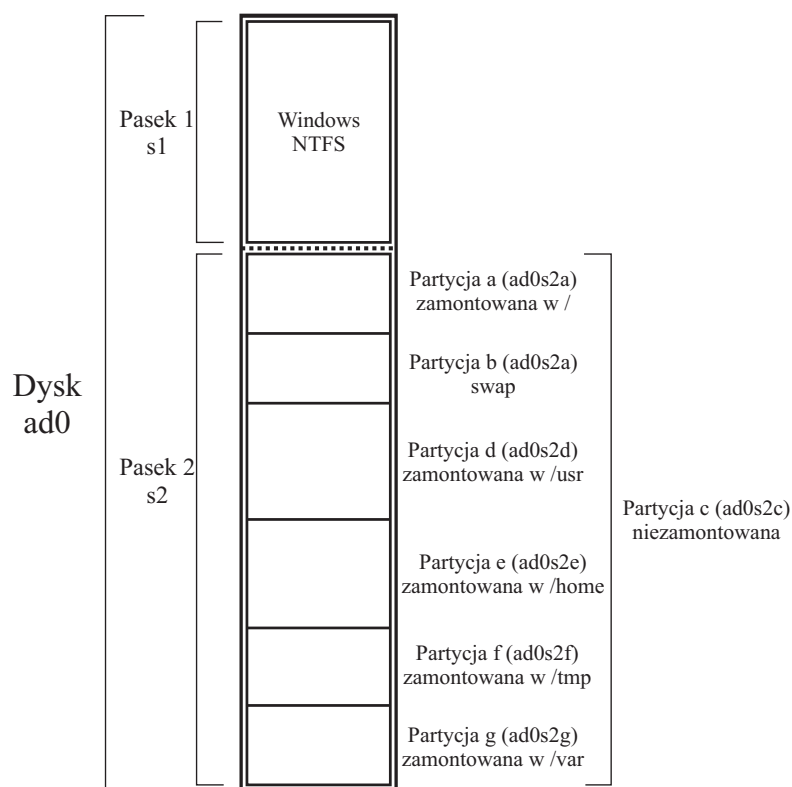
## 3.5 Administracja systemami plików w dystrybucji FreeBSD

### 3.5.1 Nazwy plików urządzeń w dystrybucji FreeBSD

Przykładową organizację dysku w systemie FreeBSD przedstawiono na rysunku 3.5.1.

Jak widać hierarchia składa się z trzech stopni: dysku fizycznego, paska (*slice*) i występujących na nim partycji. Nazwa pliku reprezentującego urządzenie składa się z czterech części:

1. Pierwsza określa typ urządzenia dyskowego. Oznaczenia najczęściej występujących przedstawiono poniżej:
  - **ad** – dysk typu ATAPI (IDE).
  - **da** – dysk typu SCSI (o dostępie bezpośrednim).
  - **acd** – napęd CDROM ATAPI (IDE).
  - **cd** – napęd CDROM SCSI.
  - **fd** – napęd dyskietek elastycznych.
2. Druga to liczba będąca numerem urządzenia danego typu zainstalowanego w systemie. Numeracja rozpoczyna się od 0.
3. Trzecia określa numer paska na dysku poprzedzony literą *s* (od *slice*). Numeracja rozpoczyna się od 1. Przyjęło się, że na pasku znajdują się zazwyczaj partycje przeznaczone dla systemów plików jednego systemu operacyjnego. Na urządzeniu fizycznym mogą znajdować się maksymalnie cztery paski, ale jeden z nich może być typu *extended*, który można następnie podzielić na paski logiczne. Numery pasków logicznych zaczynają się od 5.
4. Ostatnia to litera oznaczająca partycję w obrębie paska. Numeracja rozpoczyna się od litery *a*. Niektóre z oznaczeń dostarczają dodatkową informację o tym, co na danej partycji się znajduje. Są to litery:
  - **a** – zazwyczaj oznacza partycję zawierającą główny system plików.
  - **b** – zazwyczaj oznacza partycję, na której znajduje się plik wymiany pamięci wirtualnej.



Rysunek 3.29: Organizacja dysku w systemie operacyjnym FreeBSD.

- **c** – partycja ta reprezentuje pasek. Nie ma możliwości tworzenia na niej systemów plików. Nie posiada ona punktu montowania.
- **d** – partycja to może mieć specjalne przeznaczenie dla systemu. Można na niej zakładać systemy plików.

Pozostałe są wykorzystywane do oznaczania zwykłych partycji zawierających systemy plików.

Przykładowo przedstawiony na rysunku 3.5.1 system plików zamontowany w katalogu `/home` znajduje się na urządzeniu, które reprezentuje plik `/dev/ad0s2e`. Na podstawie nazwy pliku możemy stwierdzić, że partycja ta znajduje się na pierwszym dysku IDE (`ad`), na utworzonym na nim drugim pasku (`s2`), na partycji `e`.

### 3.5.2 Zarządzanie paskami i partycjami

System FreeBSD charakteryzuje się nieco odmiennym sposobem zarządzania przestrzenią dyskową, polegającym na podzieleniu dysków na paski (odpowiednik partycji w innych systemach), a następnie ich podziałowi na partycje, na których mogą zostać założone systemy plików. Jak wspomniano, pasek przeznaczony jest dla jednego systemu operacyjnego. Stąd w skrajnym przypadku możemy na dysku założyć jeden pasek, a następnie podzielić go na partycje i zainstalować system FreeBSD. System operacyjny może zostać zainstalowany na większej liczbie pasków. Maksymalna

liczba pasków na dysku wynosi 4. Dodanie nowego dysku do systemu wymaga w pierwszej kolejności podzielenia go na paski. Operację tę możemy przeprowadzić z wykorzystaniem instalatora systemu. Możemy uruchomić go poleceniem *sysinstall* w działającym systemie. Proces podziału będzie przebiegał identycznie, jak podczas instalacji systemu. Mamy również możliwość skorzystania z polecenia *fdisk*, które w przypadku tej rodziny systemów operacyjnych działa nieco inaczej. Podobnie wygląda kwestia podziału paska na partycje. Tu najprościej będzie skorzystać z instalatora, a w dalszej kolejności, w zależności od przeznaczenia partycji, posłużyć się odpowiednimi programami narzędziowymi.

### Zarządzanie paskami z wykorzystaniem programu *fdisk*

Założmy, że do systemu został dodany nowy dysk. Reprezentuje go plik */dev/ad1*. Jego zerowy sektor powinien zawierać kod ładujący system operacyjny, tablicę pasków oraz etykietę. Zalecaną czynnością jest wyzerowanie jego tablicy partycji. Możemy je przeprowadzić w następujący sposób:

```
1 root@messy ~ # dd if=/dev/zero of=/dev/ad1 bs=1024 count=1
```

Najprostszą czynnością jest założenie jednego paska. Wykonujemy je w następujący sposób:

```
1 root@messy ~ # fdisk -I /dev/ad1
```

O zawartości tablicy pasków możemy przekonać się uruchamiając program *fdisk*, podając jako argument nazwę pliku urządzenia. Wynik dla naszego przypadku jest następujący:

```
1 root@messy ~ # fdisk /dev/ad1
2 ***** Working on device /dev/ad1 *****
3 parameters extracted from in-core disklabel are:
4 cylinders=19767 heads=16 sectors/track=63 (1008 blks/cyl)
5
6 Figures below won't work with BIOS for partitions not in cyl 1
7 parameters to be used for BIOS calculations are:
8 cylinders=19767 heads=16 sectors/track=63 (1008 blks/cyl)
9
10 Media sector size is 512
11 Warning: BIOS sector numbering starts with sector 1
12 Information from DOS bootblock is:
13 The data for partition 1 is:
14 sysid 165 (0xa5), (FreeBSD/NetBSD/386BSD)
15   start 63, size 19925073 (9729 Meg), flag 80 (active)
16     beg: cyl 0/ head 1/ sector 1;
17     end: cyl 310/ head 15/ sector 63
18 The data for partition 2 is:
19 <UNUSED>
20 The data for partition 3 is:
21 <UNUSED>
22 The data for partition 4 is:
23 <UNUSED>
```

Jak widać, listing rozpoczyna informacje dotyczące geometrii dysku (linie 3–10). W drugiej części pojawiła się zawartość tablicy pasków (linie 12–23). Mamy założony jeden pasek. Jego identyfikator to 165 (linia 14). Rozpoczyna się on na 63 sektorze, ma rozmiar 19925073 sektorów czyli 9729 MB i jest aktywny (linia 15). Kolejne dwie linie, to opis rozmieszczenia. Na dysku nie ma więcej pasków (linie 18–23).

Informacje dotyczące zawartości tablicy pasków można uzyskać również w innym formacie. Format ten jest wykorzystywany do zapisu sposobu podziału dysku na paski, a następnie zapisaniu tej informacji programem *fdisk* do tablicy pasków. W naszym przypadku wygląda to następująco:

```

1 root@messy ~ # fdisk -p /dev/ad1
2 # /dev/ad1
3 g c19767 h16 s63
4 p 1 0xa5 63 19925073
5 a 1

```

Format pliku jest dość intuicyjny. Znak # jest znakiem komentarza, który rozpoczyna się od pierwszego znaku po nim i ciągnie do znaku przejścia do nowej linii. Każda linia, która nie jest komentarzem rozpoczyna się od etykiety, która definiuje jej znaczenie.

Linia rozpoczynająca się literą *g* służy do zdefiniowania geometrii dysku, co jest wykorzystywane do obliczeń rozmieszczenia partycji. Jej składnia to: *g cnum hnum snum*, gdzie *c* definiuje liczbę cylindrów, *h* liczbę głowic, a *s* liczbę sektorów. Kolejność definiowania jest dowolna. Linia definiująca geometrię dysku musi wystąpić przed linią definiującą geometrię paska. Liczba cylindrów nie może być większa niż 1024.

Do definiowania geometrii pasków służą linie rozpoczynające się literą *p*. Składnia linii jest następująca: *p numer typ początek rozmiar*. Wartość pola *numer* może przyjmować wartości z zakresu 1–4 i oznacza numer partycji. Najczęściej spotykanym oznaczeniem typu jest 165, co oznacza pasek wykorzystywany przez system z rodziny BSD. Wartość pola *początek* to numer sektora, od którego rozpoczyna się dany sektor. W polu tym może pojawić się znak \*, który oznacza, że pasek zaczyna się na pierwszym sektorze po ostatnim poprzedniego paska. Pole *rozmiar* przechowuje rozmiar paska. Jeśli jest to liczba niemianowana, to jednostką jest sektor. Rozmiar może zostać podany w kilobajtach, megabajtach lub gigabajtach i wówczas konieczne jest zakończenie liczby literą odpowiednio K, M lub G. Jeśli w polu tym pojawi się znak \*, to wówczas pasek będzie ciągnął się do końca dysku.

Pasek aktywny wskazuje linia rozpoczynająca się literą *a*. Po niej musi pojawić się numer paska, a więc liczba z przedziału 1–4. Linia może pojawić się w pliku konfiguracyjnym jeden raz, w dowolnym miejscu.

Opis podziału naszego dysku na dwa paski o rozmiarach po 3 GB i trzeci ciągnący się do końca dysku, z aktywną partycją numer 1 będzie następujący:

```

1 # /dev/ad1
2 g c19767 h16 s63
3 p 1 0xa5 63 3G
4 p 2 0xa5 * 3G
5 p 3 0xa5 * *
6 a 1

```

Podział zapisujemy do tablicy pasków:

```
1 root@messy ~ # fdisk -f slice_ad1 /dev/ad1
```

A następnie sprawdzamy skuteczność wprowadzonych zmian:

```
1 root@messy ~ # fdisk /dev/ad1
2 ***** Working on device /dev/ad1 *****
3 parameters extracted from in-core disklabel are:
4 cylinders=19767 heads=16 sectors/track=63 (1008 blks/cyl)
5
6 Figures below won't work with BIOS for partitions not in cyl 1
7 parameters to be used for BIOS calculations are:
8 cylinders=19767 heads=16 sectors/track=63 (1008 blks/cyl)
9
10 Media sector size is 512
11 Warning: BIOS sector numbering starts with sector 1
12 Information from DOS bootblock is:
13 The data for partition 1 is:
14 sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
15     start 63, size 6290865 (3071 Meg), flag 80 (active)
16     beg: cyl 0/ head 1/ sector 1;
17     end: cyl 96/ head 15/ sector 63
18 The data for partition 2 is:
19 sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
20     start 6290928, size 6290928 (3071 Meg), flag 0
21     beg: cyl 97/ head 0/ sector 1;
22     end: cyl 193/ head 15/ sector 63
23 The data for partition 3 is:
24 sysid 165 (0xa5),(FreeBSD/NetBSD/386BSD)
25     start 12581856, size 7343280 (3585 Meg), flag 0
26     beg: cyl 194/ head 0/ sector 1;
27     end: cyl 310/ head 15/ sector 63
28 The data for partition 4 is:
29 <UNUSED>
```

W powyższym przykładzie wykorzystano najczęściej używane opcje polecenia *fdisk*. Poniżej zamieszczono ich opis oraz przedstawiono krótko pozostałe. Argumentem polecenia może być nazwa pliku urządzenia. Jeśli zostanie on pominięty, to program pracuje na pierwszym urządzeniu dyskowym.

- **-a** – opcja zmienia jedynie aktywny pasek. Nie jest aktywna jeśli jednocześnie użyto opcji **-f**.
- **-b** *plik\_z\_kodem\_ładującym\_system* – umożliwia odczyt kodu z pliku, którego nazwa została podana jako wartość opcji. Domyślnie jest to plik */boot/mbr*.
- **-B** – opcja pozwala na dokonanie reinicjalizacji kodu inicjalizującego zapisanego w sektorze 0 dysku. Opcja jest ignorowana jeśli jednocześnie użyto opcję **-f**.
- **-f** *nazwa\_pliku\_konfiguracyjnego* – opcja pozwala na zapisanie parametrów pasków dysku w oparciu o opis zawarty w pliku. Domyślnie wykonanie polecenia spowoduje modyfikację

jedynie wskazanych pasków. Jeśli natomiast użyto opcji **-i**, to cała informacja o istniejącej konfiguracji zostanie usunięta, a na jej miejscu pojawi się zapisana w pliku. Jeśli zamiast nazwy pliku pojawi się znak **-**, to konfiguracja będzie czytana ze standardowego wejścia.

- **-i** – umożliwia inicjalizację sektora 0 dysku. Dotychczasowa konfiguracja pasków jest zaznaczana jako nieaktywna.
- **-I** – pozwala na prostą inicjalizację sektora 0 dysku, polegającą na zdefiniowaniu jednego paska na całym dysku dla systemu BSD (identyfikator 165).
- **-p** – opcja umożliwia wypisanie konfiguracji pasków na dysku. Informacja jest podawana w formacie przedstawionym w przykładzie.
- **-q** – opcja wymusza „cichą” pracę polecenia, w której niektóre komunikaty nie są wypisywane.
- **-s** – umożliwia wypisanie krótkiej informacji o rozmieszczeniu pasków na dysku.
- **-t** – przełącza tryb pracy do trybu testowego, w którym żadne zmiany nie zostaną zapisane w sektorze 0. Najczęściej jest wykorzystywana z opcją **-f** w celu sprawdzenia poprawności pliku konfiguracyjnego lub co zostanie zapisane w sektorze 0 w przypadku jego wykorzystania.
- **-u** – umożliwia pracę w trybie interaktywnym. Jest ignorowana jeśli użyto opcji **-f**.
- **-v** – włącza tryb gadatliwy. Pożądana w przypadku użycia opcji **-f**, gdyż raportuje wprowadzane zmiany.
- **-1234** – opcja umożliwia wskazanie pojedynczego paska, na którym polecenie będzie pracować. Ignorowana w przypadku użycia opcji **-f**.

### Zarządzanie paskami z wykorzystaniem programu *sysinstall*

Prostszym sposobem dokonania podziału dysku na paski jest skorzystanie z programu *sysinstall*. Po jego uruchomieniu wybieramy typ instalacji *Custom*, następnie punkt 3 *Partitions*. W kolejnym okienku instalatora zaznaczamy urządzenie, na którym mają zostać wydzielone paski i przechodzimy do edytora, który dla przypadku naszego dysku przedstawiono na rysunku 3.30.

Konfiguracja, jaką zastaliśmy na dysku */dev/ad1* została zbudowana ostatnim, przykładowym uruchomieniem programu *fdisk* wykorzystującym plik opisujący podział dysku na paski. Została ona wczytana do struktur pamięci programu, dzięki czemu wprowadzone przez nas zmiany nie zostaną zapisane na dysku bez wydania odpowiedniej komendy. Lista dostępnych komend znajduje się w dolnej części okna. W górnej mamy przedstawiony obraz podziału dysku. Opisany został w tabelce, posiadającej 8 kolumn. Znaczenie kolumn jest następujące:

1. *Offset* – numer sektora, od którego zaczyna się dany pasek.
2. *Size* – rozmiar paska w sektorach. Domyślnie podawany w sektorach. Istnieje możliwość zmiany na KB, MB lub GB.
3. *End* – numer ostatniego sektora paska.
4. *Name* – nazwa pliku reprezentującego urządzenie.
5. *PType* – typ paska. Oznaczenia jak w programie *fdisk*: 8 oznacza pasek ogólnego przeznaczenia, 12 diagnostyczny lub przechowujący informację o konfiguracji.



```

Disk name: ad1          DISK Partition Editor
DISK Geometry: 19767 cyls/16 heads/63 sectors = 19925136 sectors (9729MB)

Offset      Size(ST)      End      Name  PType  Desc  Subtype  Flags
-----
      0         63         62      -    12   unused      0
     63    6290865    6290927   ad1s1    8   freebsd    165
    6290928    6290928    12581855   ad1s2    8   freebsd    165
   12581856    7343280    19925135   ad1s3    8   freebsd    165
  19925136         744   19925879      -    12   unused      0

The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry    C = Create Slice
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable    I = Expert m.
T = Change Type        U = Undo All Changes    W = Write Changes   Q = Finish

Use F1 or ? to get more help, arrow keys to select.

```

Rysunek 3.30: Okno instalatora systemu będące edytorem pasków dla wskazanego dysku. Początkowa zawartość dysku pochodzi z jego sektora 0.

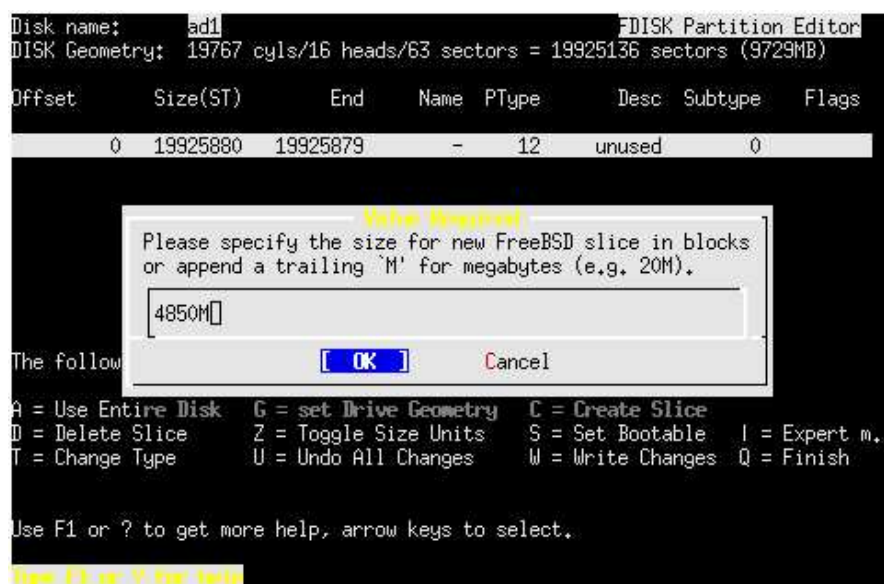
6. *Desc* – opisuje przeznaczenie paska w oparciu o wartość jego identyfikatora.
7. *Subtype* – identyfikator typu paska.
8. *Flags* – dodatkowy opis paska. Najczęściej oznaczenia spotykane to: = – pasek poprawnie dopasowany, A – pasek aktywny, R – pasek zawiera główny system plików.

Zmienimy istniejącą konfigurację wprowadzając podział na dwa paski o zbliżonych rozmiarach. W tym celu usuwamy istniejące, najeżdżając paskiem podświetlenia na paski *ad1s1–ad1s3* i naciskając klawisz *D*. Dla utworzenia nowego paska wybieramy klawisz *C*, podajemy jego rozmiar w blokach (liczba niemianowana) lub megabajtach (wartość zakończona literą M). Wartość domyślna podawana jest w blokach i w tym przypadku jest równa całej pojemności dysku. Ponieważ dysk ma pojemność 9.7 GB, stąd proponowany rozmiar paska został ustalony 4850 MB (rysunek 3.31). Następnie należy podać identyfikator. Wartość domyślna to 165. W identyczny sposób tworzymy drugi pasek, tym razem na pozostałej części dysku.

Po dokonaniu podziału dysku na paski, zaproponowaną konfigurację należy zapisać. Wybieramy klawisz *W*. Istotność przeprowadzanych operacji powoduje, że program wymaga jeszcze ich potwierdzenia (wybieramy przycisk *Ok* zamiast domyślnego *No*). Następnie pojawia się okno umożliwiające zainstalowanie programu ładującego system operacyjny. W naszym przypadku wybieramy opcję *None*, która nie spowoduje jego instalacji. Na koniec pojawia się informacja o zapisaniu stworzonej konfiguracji na dysku, co efektywnie kończy proces podziału dysku na paski.

Opcje edytora pasków dostępnego w programie instalatora są następujące:

- **A** – umożliwia założenie paska na całym dysku.
- **D** – usuwa wskazany pasek.



Rysunek 3.31: Okienko dialogowe umożliwia podanie rozmiaru tworzonego paska. W tle widoczny opis dysku, na którym nie zdefiniowano żadnego paska.

- **T** – służy zmianie typu paska. Do najpopularniejszych należą: 165 – pasek wykorzystywany przez FreeBSD, 131 – pasek przeznaczony dla systemu plików z rodziny ext, 130 – na pasku znajdować się będzie plik wymiany pamięci wirtualnej systemu Linux.
- **G** – umożliwia ustalenie efektywnej geometrii dysku.
- **Z** – pozwala na zmianę jednostek, w których operuje program.
- **U** – cofa wszystkie wprowadzone zmiany.
- **C** – umożliwia utworzenie nowego paska.
- **S** – czyni pasek aktywnym.
- **W** – zapisuje wprowadzone zmiany.
- **Q** – kończy pracę edytora i powoduje powrót do poprzedniego okna instalatora.
- **I** – przełącza w tryb eksperta.

### Zarządzanie paskami z wykorzystaniem programu *sysinstall*

W najprostszy sposób podziału utworzonych pasków na partycje można dokonać wykorzystując program *sysinstall*. Po jego uruchomieniu wybieramy instalację typu *Custom*, a następnie pozycję *Label* (Label allocated disk partitions). W kolejnym okienku dialogowym należy zaznaczyć dysk, którego paski mają zostać podzielone na partycje. Kolejne okienko, to edytor etykiet, który przedstawiono na rysunku 3.32.

W górnej części okna mamy możliwość wyboru paska, który zostanie podzielony na partycje. Pasek opisany jest nazwą pliku reprezentującego urządzenie, na którym pasek się znajduje, nazwą



Rysunek 3.32: Okno edytora pasków dla wskazanego urządzenia dyskowego.

pliku urządzenia reprezentującego pasek oraz dodatkowo ilością dostępnego na nim miejsca. Część środkowa daje podgląd istniejących partycji na wybranym pasku. Tabelka zapisana jest w dwóch częściach po cztery kolumny. Kolumny oznaczają:

1. *Part* – nazwę pliku urządzenia reprezentującego partycję.
2. *Mount* – punkt montowania partycji w drzewie katalogów.
3. *Size* – rozmiar partycji.
4. *Newfs* – rodzaj istniejącego na partycji systemu plików.

Część dolna zawiera informację o możliwych do wykonania czynnościach i klawiszach je uruchamiających.

Na urządzeniu *ad1* założymy trzy partycje, na których założone zostaną zwykłe systemy plików. Dwie o zbliżonych rozmiarach pojawią się na pasku pierwszym. Trzecia wykorzysta całe miejsce dostępne na drugim pasku. W tym celu, korzystając z klawiszy sygnalizacyjnych wybieramy w górnej części okna właściwy pasek. Następnie naciskamy klawisz *C* (Create). Pojawia się okienko dialogowe, w którym wpisujemy rozmiar tworzonej partycji. Rozmiar możemy podać w blokach (liczba niemianowana), megabajtach (M), gigabajtach (G) lub cylindrach (C). W naszym przypadku będzie to 2425M. Po zatwierdzeniu przyciskiem *Ok*, pojawia się kolejne okno, w którym dokonujemy wyboru rodzaju systemu plików, który zostanie założony na tworzonej partycji. Do wyboru mamy zwykły system plików (UFS) lub partycję dla pliku wymiany pamięci wirtualnej. Jeśli wybrany zostanie zwykły system plików, w kolejnym okienku dialogowym podajemy punkt montowania systemu plików w drzewie katalogów. Wskazany katalog nie musi istnieć, zostanie automatycznie utworzony. Identycznie postępujemy w przypadku drugiej partycji. Tu zgadzamy się z zaproponowanym rozmiarem, gdyż jest to całe wolne miejsce. Podajemy inny punkt montowania. Po wykonaniu tych czynności ilość dostępnego miejsca na pasku pierwszym

wynosi 0. Przenosimy się na pasek drugi i trzecią partycję zakładamy korzystając z wartości domyślnych, proponowanych przez program instalatora. Wyjątek stanowi konieczność podania kolejnego punktu montowania.



Rysunek 3.33: Okno edytora pasków dla wskazanego urządzenia dyskowego.

Nowa konfiguracja została przeprowadzona w pamięci programu. Jeśli jest ona poprawna, należy ją zapisać korzystając z klawisza W (Write). Spowoduje to zapisanie tablicy partycji na dysku oraz założenie na utworzonych partycjach systemów plików. Jeśli wskazane jako punkty montowania katalogi nie istniały, to zostaną utworzone. Nie zostaną jednak dokonane żadne wpisy w pliku */etc/fstab*. Musimy je dokonać ręcznie. W naszym przypadku będą one następujące:

1	/dev/ad1s1d	/scratch1	ufs	rw	0	0
2	/dev/ad1s1e	/scratch2	ufs	rw	0	0
3	/dev/ad1s2d	/scratch3	ufs	rw	0	0

Systemy plików stają się dostępne po ich zamontowaniu:

1	root@messy ~ # mount -a
2	root@messy ~ # df
3	Filesystem 1K-blocks Used Avail Capacity Mounted on
4	/dev/ad0s1a 4058062 483040 3250378 13% /
5	devfs 1 1 0 100% /dev
6	/dev/ad0s1e 4058062 63982 3669436 2% /home
7	/dev/ad0s1f 2026030 240834 1623114 13% /tmp
8	/dev/ad0s1d 8122126 4689500 2782856 63% /usr
9	/dev/ad0s1g 4058062 4310 3729108 0% /var
10	/dev/ad1s1d 2400238 4 2208216 0% /scratch1

11	/dev/ad1s2d	4836734	4 4449792	0%	/scratch3
12	/dev/ad1s1e	2399718	4 2207738	0%	/scratch2

Na zakończenie krótki opis dostępnych opcji służących do zarządzania partycjami:

- *C* – umożliwia utworzenie nowej partycji na wskazanym pasku.
- *N* – pozwala na modyfikację opcji zakładanego systemu plików.
- *T* – służy do wyboru rodzaju zakładanego systemu plików.
- *D* – usuwa wskazaną partycję.
- *Q* – pozwala wyjść z programu bez zapisania wprowadzonych zmian.
- *U* – cofa wprowadzone zmiany.
- *M* – pozwala zdefiniować punkt montowania zakładanego systemu plików.
- *A* – umożliwia wprowadzenie domyślnych wartości atrybutów systemu plików.
- *R* – pozwala na usunięcie wybranej partycji i połączenie pozostałych tak, aby zajmowały spójny obszar dysku.
- *Z* – opcja do strojenia zakładanego systemu plików, umożliwia nadanie własnych wartości atrybutom systemu plików.
- *W* – opcja powoduje zapisanie stworzonej konfiguracji partycji i przejście do następnego okna programu instalatora.

### 3.5.3 System plików UFS

Podstawowym systemem plików wykorzystywanym w systemie FreeBSD jest Fast File System (FFS), będący bezpośrednim potomkiem systemu plików z systemu BSD4.4. FFS znany jest też jako UFS (od *UNIX File System*), a wiele narzędzi systemowych nadal określa partycję FFS mianem UFS. System powstał w połowie lat osiemdziesiątych XX wieku. Jego twórcami są Marshall Kirk McKusick i Bill Joy. Jego pierwszy opis można znaleźć w [5]. Posiada wiele ciekawych rozwiązań, jak np. tzw. miękkie uaktualnienia (*soft updates*), które w przypadku awarii zasilania zapewniają jego wewnętrzną spójność. Nie gwarantują jednak odzyskania danych.

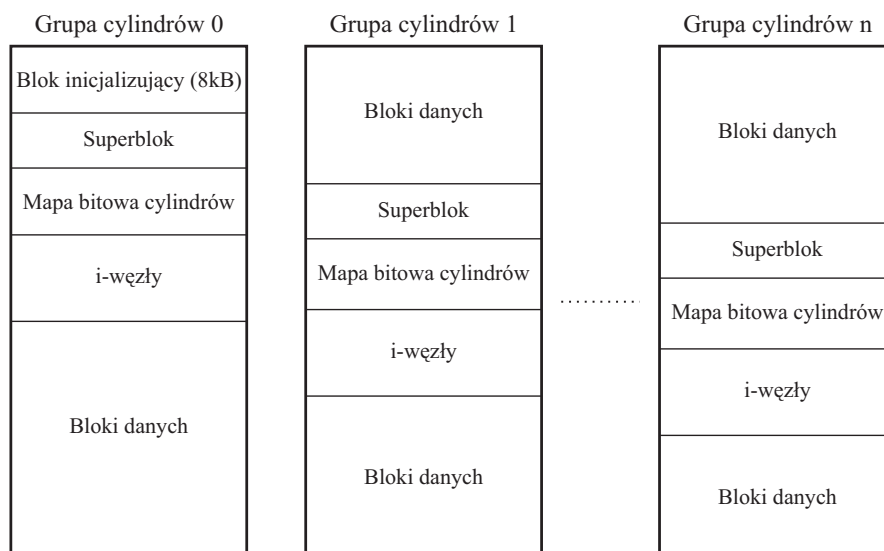
#### Budowa wewnętrzna

System plików *UFS* składa się z tzw. grupy cylindrów, które tworzone są w momencie zakładania systemu plików. Grupa cylindrów to jeden lub więcej cylindrów dyskowych położonych obok siebie, tworzących ciągłą przestrzeń adresową. Grupy cylindrów zostają następnie podzielone na adresowalne bloki wykorzystywane do zarządzania strukturą i informacją przechowywaną w grupach cylindrów. Wyróżnia się cztery typy bloków, z których każdy ma specyficzną funkcję w systemie plików:

1. *Blok inicjalizujący (bootujący).*
2. *Superblok* zwany również blokiem identyfikacyjnym.
3. *i-węzeł.*

## 4. Blok danych.

Jak widać rodzaje bloków oraz ich funkcje są identyczne z występującymi w systemie plików *s5*. Całkowicie różne jest jednak ich rozmieszczenie na dysku. Wykorzystując cechy charakterystyczne geometrii ma podnosić jego wydajność. Schematycznie budowę wewnętrzną przedstawiono na rysunku 3.34.



Rysunek 3.34: Schemat budowy wewnętrznej systemu plików UFS.

Blok inicjalizujący, inaczej bootujący, zawiera kod wykorzystywany do inicjalizacji systemu operacyjnego. Jeśli dana partycja nie jest wykorzystywana do inicjalizacji systemu, to blok ten jest pusty. Blok ten występuje jedynie w pierwszej grupie cylindrów (grupa 0) i zajmuje pierwsze 8 kB.

Superblok przechowuje wartości podstawowych atrybutów systemu plików. Do podstawowych należą:

- Rozmiar i status systemu plików.
- Etykiety (nazwy) systemu plików i dysku.
- Rozmiar bloku logicznego wykorzystywanego w systemie plików.
- Czas ostatniej zmiany zawartości systemu plików.
- Rozmiar grupy cylindrów.
- Stan systemu plików (czysty, aktywny, stabilny).
- Ścieżka dostępu do ostatniego punktu montowania.

Blok identyfikujący znajduje się na początku paska i jest powielony w każdej grupie cylindrów. Istotność przechowywanych w superbloku danych powoduje, że wszystkie jego kopie są tworzone już w momencie zakładania systemu plików. Kopie znajdują się jednak w różnych miejscach

każdej grupy cylindrów. Różną odległość położenia od początku grupy cylindrów uzyskuje się umieszczając na ich początku (za wyjątkiem pierwszej grupy cylindrów) bloki danych w różnych ilościach. Dodatkowo, jeśli dany system plików znajduje się na fizycznie kilku talerzach dysku, to superbloki rozmieszcza się tak, aby znajdowały się po jednym na każdym talerzu. Stanowi to przyczynek do podniesienia poziomu bezpieczeństwa systemu plików. Superblok znajdujący się w pierwszej grupie cylindrów zawiera dodatkowo blok z sumarycznym opisem stanu systemu plików. Jego treść stanowią m.in. listy z numerami wolnych i-węzłów oraz numerami wolnych bloków danych. Zwiększają one wydajność systemu plików. Nie muszą być przechowywane, a co za tym idzie uaktualniane we wszystkich superblokach. W razie awarii informację tę można odtworzyć przeglądając system plików.

I-węzły zawierają podstawową informację opisującą plik, za wyjątkiem jego nazwy, która przechowywana jest w katalogu. Nazwę odszukuje się za pomocą numeru i-węzła, które oprócz rodzaju pliku tworzą wpis w katalogu. Rozmiar i-węzła w systemie plików UFS jest stały i wynosi 128 B. Do podstawowych, przechowywanych w nim informacji należy zaliczyć:

- Rodzaj pliku.
- Prawa dostępu do pliku.
- Liczbę jego dowiązań twardych, a więc nazw pod którymi występuje.
- Numer identyfikacyjny użytkownika będący jego właścicielem indywidualnym oraz numer identyfikacyjny grupy, które jest jego właścicielem grupowym.
- Rozmiar pliku w B.
- 15-to pozycyjną tablicę zawierającą 12 adresów bezpośrednich wskazujących bloki zawierające informację zapisaną do pliku oraz adresy od jednokrotnie do trzykrotnie pośrednich.
- Czasy: ostatniego dostępu do pliku, ostatniej jego modyfikacji oraz utworzenia pliku.

W stosunku do systemu plików *sf5* została lekko zmodyfikowana tablica rozmieszczenia pliku na dysku. Zawiera ona bowiem 12 zamiast 10 adresów bezpośrednich. Wpływa to na zwiększenie efektywności dostępu do niewielkiej klasy plików (o rozmiarach 11 i 12 bloków danych) praktycznie nie zwiększając rozmiaru największego, możliwego do przechowania pliku.

Bloki danych zajmują pozostały obszar w obrębie grupy cylindrów. W nich zapisywana jest informacja przechowywana w plikach i katalogach. Istotną różnicą w stosunku do innych systemów plików jest możliwość współdzielenia bloku danych przez więcej niż jeden plik. Pozwala to na efektywne gospodarowanie miejscem w blokach danych przechowujących „końce” plików, które w tradycyjnym podejściu jest niewykorzystane, bowiem rzadko się zdarza, aby rozmiar pliku stanowił całkowitą wielokrotność rozmiaru bloku danych. Domyślny rozmiar bloku danych wynosi 8192 B, zaś rozmiar fragmentu, nazywanego także blokiem logicznym wynosi 1024 B. W przypadku partycji o rozmiarze ponad 1 GB zaleca się stosowanie większych rozmiarów bloków danych, np. 16384 lub 32768 B. Ze względów wydajnościowych rozmiar bloku logicznego powinien wynosić 1/8 rozmiaru bloku danych.

### Mechanizm miękkich poprawek (*softupdates*)

Istotną sprawą jest kwestia dziennika systemu plików. UFS został wyposażony w mechanizm tzw. miękkich poprawek (*softupdates*). Jest on w stanie zagwarantować jedynie spójność systemu plików, który pracował np. w chwili zaniku zasilania. Dodatkową korzyścią jest zwiększenie wydajności systemu plików, zwłaszcza dla operacji tworzenia i usuwania plików. Omówmy krótko działanie tego mechanizmu.

Generalnie istnieją dwie metody zapisu modyfikowanych metadanych systemu plików na dysk. Pierwsza polega na zapisywaniu metadanych w sposób synchroniczny. Jeśli przykładowo zmianie uległa zawartość katalogu, to system czeka aż wszystkie zmiany metadanych go opisujących zostaną zapisane na dysku. Dane plików, znajdujące się w buforach plikowych zostaną zapisane na dysku później, w sposób asynchroniczny. Przy takim podejściu metadane są zawsze spójne. Przykładowo, operacja tworzenia pliku utworzy go kompletnego lub nie utworzy wogóle. Prostota podejścia implikuje prostą i stabilną implementację. Wadą jest wydłużenie czasu operacji na metadanych.

Drugim rozwiązaniem jest asynchroniczny zapis metadanych. W tym podejściu zapis metadanych odbywa się również z wykorzystaniem pamięci podręcznej, jednak jest on wykonywany naprzemiennie z zapisem danych pliku. Podejście to zapewnia krótszy czas operacji wymagających istotnych zmian metadanych systemu plików. Również w tym przypadku implementacja jest prosta i nie powinna stanowić źródła błędów. Wadą jest to, iż można znaleźć takie momenty czasu, w których stan systemu plików jest niespójny. Zostanie on utrwalony na dysku, jeśli w nich wystąpi zanik zasilania. Może zdarzyć się, że na dysku zostaną zapisane nowe bloki danych pliku, natomiast nie zostaną uaktualnione opisujące go metadane. Wówczas dane zostaną utracone, gdyż w systemie plików brak jest dostatecznej ilości informacji do ich odzyskania.

Najprostszym rozwiązaniem tego problemu jest tzw. dziennikowanie brudnego regionu (*dirty region logging*), zwane niekiedy wprost dziennikowaniem choć, jak się przekonamy nie całkowicie poprawnie. Polega ono na zapisywaniu zmian metadanych w sposób synchroniczny, ale jedynie na małym obszarze dysku. W późniejszym czasie zostaną one przeniesione w odpowiednie miejsce. Mały rozmiar obszaru, w którym zapisuje się dane oraz zajmowanie przez niego spójnego fragmentu dysku implikują, nawet w przypadku operacji wymagających istotnych zmian, krótki czas wykonania (jest on krótszy od czasu zmian zapisywanych synchronicznie, bezpośrednio w metadanych systemu plików). Wadą tego rozwiązania jest dwukrotne zapisywanie zmian metadanych, co w przypadku operacji istotnie je modyfikujących może wydłużyć czas ich wykonania. Jednak w przypadku awarii, znane są zmiany w metadanych systemu plików, co pozwala na podrowadzenie go do stanu spójności.

Nieco inne rozwiązanie zostało zaproponowane w systemie plików UFS. Wszystkie bierzące zmiany w metadanych przechowywane są w pamięci i zapisywane na dysk w odpowiedniej kolejności (*ordered meta-data updates*). Kolejność jest istotna jeśli przykładowo późniejsze zmiany metadanych danej operacji obejmują te wcześniejsze i nie zostały zapisane na dysku. Przechowywane w pamięci informacje są porządkowane, a następnie usuwane nadmiarowe. Dla zmian w blokach danych, porządkuje się je tak, aby zminimalizować ruchy głowicy dysku, konieczne do ich wprowadzenia. W przypadku awarii, wykonuje się tzw. przewinięcie dziennika, przyjmując iż operacje, które nie zakończyły się zapisaniem informacji na dysku nie miały miejsca. Uwzględniając częstotliwość zapisu informacji do systemu plików, można przyjąć, że zostanie osiągnięty stan spójności systemu plików sprzed 30–60 sekund. Zastosowany algorytm zapewnia, że wykorzystywane bloki danych i i-węzły będą zaznaczone zgodnie z ich stanem zapisanym w odpowiednich mapach bitowych. Jedyne błędy jakie może pojawić się w wyniku awarii polega na zaznaczeniu jakiegoś zasobu jako wykorzystanego w momencie, kiedy będzie on wolny. Sytuacja ta jest jednak możliwa do wychwycenia przez program *fsck*. Metoda ta posiada jednak swoje wady. Należy do nich zaliczyć relatywnie skomplikowany kod oraz wysokie zużycie pamięci operacyjnej.

### Zakładanie systemu plików

W systemach FreeBSD, do zakładania oraz inicjalizacji przed pierwszym użyciem systemu plików służy polecenie *newfs*. W najprostszym przypadku wymaga użycia jedynie argumentu, którym jest nazwa pliku reprezentującego urządzenie, na którym ma zostać założony lub zainicjalizowany



system plików. W przypadku inicjalizacji system plików nie może być zamontowany z opcją do zapisu. Przykład utworzenia nowego systemu plików na pierwszej partycji (*d*), pierwszego paska (*s1*) urządzenia *ad1* przedstawiono poniżej:

```
1 root@messy ~ # newfs /dev/ad1s1d
2 /dev/ad1s1d: 2425.0MB (4966400 sectors) block size 16384, fragment size 2048
3     using 14 cylinder groups of 183.77MB, 11761 blks, 23552 inodes.
4 super-block backups (for fsck -b #) at:
5     160, 376512, 752864, 1129216, 1505568, 1881920, 2258272, 2634624, 3010976,
6     3387328, 3763680, 4140032, 4516384, 4892736
```

W liniach 2 oraz 3 zostały podane wartości podstawowych atrybutów utworzonego systemu plików. Zajmuje on 4966400 sektorów co daje 2425.0 MB pojemności. Rozmiar bloku danych wynosi 16384 B, zaś rozmiar bloku logicznego (fragmentu) 2048 B (czyli zalecaną 1/8). System plików tworzy 14 grup cylindrów. Każda ma pojemność 183.77 MB, posiada 11761 bloków danych oraz 23552 i-węzły. Jak łatwo policzyć, jeden i-węzeł przypada na 8192 B bloków danych. Linie 4, 5 oraz 6 zawierają numery 14 sektorów, po jednym w każdej grupie sektorów, zawierające kopie superbloku. Mogą one zostać wykorzystane przez program *fsck* w celu uzyskania informacji o wartościach atrybutów systemu plików, jeśli superblok podstawowy uległ uszkodzeniu. Położenie pierwszej kopii superbloku jest ustalone i w pierwszej wersji systemu UFS jest to cylinder 32, w wersji drugiej cylinder 160. Może się jednak zdarzyć, że ulegnie on uszkodzeniu. Stąd konieczność pamiętania położenia pozostałych kopii.

W zdecydowanej większości przypadków, system plików UFS zakłada się z domyślnymi wartościami atrybutów. Zdarzają się jednak sytuacje, w których konieczne jest użycie innych, specyficznych wartości lub oprócz założenia wymagane jest wykonanie innych czynności. Polecenie *newfs* posiada wiele opcji, spośród których do najważniejszych należy zaliczyć:

- **-E** – opcja umożliwia usunięcie zawartości dysku przed założeniem nowego systemu plików. Obszar zawierający kod inicjalizujący system operacyjny nie zostanie zmieniony. Ze względu na wykonywane operacje zapisu operacja jest zazwyczaj długotrwała.
- **-J** – opcja włącza dziennikowanie systemu plików prowadzone przez system *gjournal*.
- **-L** *etykieta wolumenu* – wartością opcji jest etykieta, która zostanie przypisana tworzonemu systemowi plików. Pominięcie opcji powoduje, że etykieta ma postać napisu pustego.
- **-N** – użycie opcji spowoduje wypisanie wartości atrybutów systemu plików przed jego założeniem.
- **-O** *typ systemu plików* – polecenie *newfs* umożliwia założenie wersji 1 oraz 2 systemu plików UFS. Domyślnie zakładana jest wersja 2. Wartość opcji pozwala określić wersję.
- **-U** – opcja włącza tzw. miękkie poprawki (*soft updates*) systemu plików. Jest to technika będąca alternatywą dla dziennikowania. Polega na odpowiednim uporządkowaniu operacji zapisu, a nie na tworzeniu kopii zapisywanych metadanych oraz ewentualnie danych w pliku dziennika. Nie gwarantuje odzyskania utraconych danych w razie awarii ale zapewnia spójność informacji przechowywanej w systemie plików. Pozwala na zamontowanie systemu plików zaraz po awarii systemu bez konieczności jego uaktualniania informacjami zapisanymi w dzienniku.

- **-a** *maksymalna\_liczba\_bloków\_w\_ciagu* – wartość opcji (opisywana jako *maxcontig*) oznacza maksymalną liczbę bloków dyskowych położonych koło siebie, które zostaną zwolnione przed wymuszaniem rotacyjnego opóźnienia. Wartość domyślna to 16. Wartość atrybutu podaje program *dumpfs*, zaś zmiana jego wartości jest możliwa w pracującym systemie plików programem *tunefs*.
- **-b** *rozmiar\_bloku* – opcja pozwala ustalić rozmiar bloku danych (bloku fizycznego) systemu plików. Podaje się go w bajtach. Wartość domyślna to 16384, a najmniejsza możliwa 4096. Rozmiar musi być potęgą liczby 2. Ze względów wydajnościowych wskazane jest, aby stosunek rozmiaru bloku do rozmiaru fragmentu (bloku logicznego) wynosił 8:1.
- **-c** *liczba\_bloków\_w\_grupie\_cylindrów* – opcja pozwala określić liczbę bloków danych przypadającą na grupę cylindrów. Wartość domyślna jest obliczana na podstawie wartości innych atrybutów, a w szczególności rozmiaru bloku danych oraz liczby bajtów danych opisywanych przez jeden i-węzeł.
- **-d** *maksymalny\_rozmiar\_obszaru* – podobnie jak inne systemy plików, UFS do przechowywania dużych plików wykorzystuje tzw. obszary zbudowane przez znajdujące się obok siebie na dysku bloki danych. Wartość opcji określa maksymalny rozmiar obszaru wykorzystywanego przez zakładany system plików, wyrażony w liczbie bloków danych. Wartość domyślna wynosi 16.
- **-e** *maxbpg* – szczególnie istotne dla wydajności systemu plików jest umieszczenie bloków danych przechowujących dany plik możliwie fizycznie blisko i-węzła opisującego ten plik. W przypadku systemu plików UFS granicą jest grupa cylindrów. Opcja pozwala określić maksymalną liczbę bloków danych alokowanych dla danego pliku, po przekroczeniu której alokowane będą bloki z innych grup cylindrów. Wartością domyślną jest 1/4 bloków dostępnych w każdej grupie cylindrów, co w praktyce oznacza, iż dane z pliku o rozmiarze większym niż 1/4 łącznego rozmiaru bloków danych w grupie cylindrów, będą zapisane w więcej niż jednej grupie cylindrów.
- **-f** *rozmiar\_fragmentu* – opcja pozwala określić rozmiar w bajtach fragmentu bloku danych (tzw. fragmentu logicznego), czyli najmniejszej jednostki przechowującej dane systemu plików. Jak wspomniano, wartość zalecana to 1/8 rozmiaru bloku danych. Wartości dopuszczalne należą do przedziału od 1/8 rozmiaru bloku do rozmiaru bloku danych.
- **-g** *spodziewany\_średni\_rozmiar\_pliku* – opcja pozwala określić spodziewany, średni rozmiar pliku przechowywanego w systemie plików. Wartość ta pozwala wyznaczyć wartości innych parametrów systemu plików, jak np. liczbę bajtów bloków danych przypadającą na jeden i-węzeł.
- **-h** *spodziewana\_średnia\_liczba\_plików\_w\_katalogu* – podobnie, jak średni rozmiar pliku w systemie plików, pozwala określić zbliżone do optymalnych wartości atrybutów systemu plików.
- **-i** *liczba\_bajtów\_na\_i-węzeł* – opcja ta określa „gęstość” i-węzłów w systemie plików, czyli liczbę bajtów danych opisywanych przez jeden i-węzeł. Domyślnie jeden i-węzeł jest tworzony dla czterech fragmentów (bloków logicznych). Ponieważ każdy plik jest opisany przez jeden i-węzeł, wartość ta jest powiązana ze spodziewanym, średnim rozmiarem pliku. Stąd im rozmiar ten jest większy, w systemie zostanie utworzonych mniej i-węzłów i odwrotnie.
- **-l** – opcja pozwala na obsługę praw dostępu modelu MAC w zakładanym systemie plików.

- **-m** *obszar\_pamięci\_dla\_użytkowników* – wartością opcji jest minimalna ilość pamięci, wyrażona w procentach pojemności systemu plików, która musi pozostać wolna w systemie plików. Nie będzie ona dostępna dla zwykłych użytkowników, a jedynie dla administratora systemu. Może ona być konieczna do pracy niektórych programów narzędziowych, jak np. *fsck*.
- **-n** – użycie opcji spowoduje, że w systemie plików nie zostanie utworzony katalog *.snap*. Jest on wykorzystywany podczas wykonywania kopii migawkowej systemu plików.
- **-o** *sposób\_optymalizacji* – wartości opcji to **space** lub **time**. Opcja umożliwia wybór sposobu optymalizacji. Pierwsza z nich umożliwia ograniczenie fragmentacji systemu plików poprzez alokowanie bloków danych dla danych jednego pliku możliwie blisko siebie, co jednak wydłuża czas wykonania procedury alokacji. Druga przeciwnie, alokuje szybko, ale bloki danych niekoniecznie położone w bezpośredniej bliskości. Domyślnie wybór zależy od rozmiaru minimalnego, zarezerwowanego obszaru. Wartość poniżej 8% oznacza optymalizację typu **space**. Dla wartości równej lub większej 8% przyjmuje się optymalizację typu **time**.
- **-p** *oznaczenie\_partycji* – to opcja kosmetyczna. Umożliwia ona rozbitcie pełnej nazwy pliku partycji na część opisującą pasek oraz literę oznaczającą partycję. Przykładowo, polecenie `mkfs /dev/ad1s1d` jest równoważne poleceniu `mkfs -p d /dev/ad1s1`.
- **-r** *rozmiar\_obszaru\_zarezerwowanego* – opcja umożliwia podanie rozmiaru obszaru na końcu partycji, na którym nie będzie założony system plików. Obszar ten może być wykorzystany przykładowo przez system *geom*. Wartość podaje się w liczbie cylindrów. Domyślnie wynosi 0.
- **-s** *rozmiar\_systemu\_plików* – wartością opcji jest rozmiar systemu plików podawany w sektorach. Wartością domyślną jest rozmiar partycji pomniejszony o rozmiar obszaru zarezerwowanego, zdefiniowanego wartością opcji **-r**. Podanie wartości 0 jest równoważne użyciu wartości domyślnej. Nie jest możliwe podanie wartości większej od domyślnej, gdyż system plików nie może wykorzystywać cylindrów przeznaczonych dla obszaru zarezerwowanego.
- **-S** *rozmiar\_sektora* – opcja pozwala zmienić wartość będącą rozmiarem sektora dysku, odczytaną z urządzenia. Jest stosowana sporadycznie. Jako przykład można podać konieczność stworzenia systemu plików, którego obraz ma docelowo pracować na urządzeniu o innej geometrii, niż to na którym jest tworzony. Zmiana wartości domyślnej jest niebezpieczna dla programu *fsck*, gdyż nie będzie on w stanie odnaleźć kopii superbloków, jeśli podstawowy uległ uszkodzeniu.

### Listowanie wartości atrybutów systemu plików

Do listowania wartości atrybutów systemu plików najlepiej użyć polecenia *dumpfs*.

```

1 [root@messy ~]# dumpfs /dev/ad15s1d
2 magic    19540119 (UFS2) time    Thu Mar 31 16:58:30 2011
3 superbloc location    65536 id      [ 4d949696 a95fd3d1 ]
4 ncg      4 size       262144 blocks 253815
5 bsize    16384 shift   14 mask    0xffffc000
6 fsize    2048 shift   11 mask    0xfffff800
7 frag     8 shift     3 fsbtodb  2
8 minfree  8% optim    time    symlinklen 120

```

```

9  maxbsize 16384  maxbpg  2048    maxcontig 8    contigsumsize 8
10 nbfree  31724  ndir    2      nifree  65788  nffree  21
11 bpg     8193  fpg     65544  ipg     16448  unrefs  0
12 nindir  2048  inopb   64     maxfilesize 140806241583103
13 sbsize  2048  cgsize  12288  csaddr  2112  cssize  2048
14 sblkno  40    cblkno  48     iblkno  56    dblkno  2112
15 cgrotor 0     fmod    0      ronly   0      clean   0
16 avgfpdir 64   avgfilesize 16384
17 flags    soft-updates
18 fsmnt    /scratch1
19 volname          swuid    0
20
21 cs[].cs_(nbfree,ndir,nifree,nffree):
22      (7926,2,16444,21) (7934,0,16448,0) (7934,0,16448,0) (7930,0,16448,0)
23 blocks in last group 8189
24
25
26 cg 0:
27 magic   90255  tell    18000  time     Thu Mar 31 16:58:30 2011
28 cgx     0      ndblk   65544  niblk    16448  initiblk 128    unrefs  0
29 nbfree  7926  ndir    2      nifree  16444  nffree  21
30 rotor   0     irotor  0      frotor   0
31 frsum    0     0      0      0      0      0      3
32 sum of frsum: 21
33 clusters 1-7: 0      0      0      0      0      0      0
34 clusters size 8 and over: 1
35 clusters free: 267-8192
36 inodes used:  0-3
37 blks free:    2113-2119, 2121-2127, 2129-65543
38
39 .....

```

### Zmiana wartości atrybutów systemu plików

Programem służącym do zmiany wartości dynamicznych atrybutów systemu plików UFS jest program *tunefs*. Wprowadzanie jakichkolwiek zmian jest możliwe wówczas, gdy system plików nie jest zamontowany lub jest zamontowany w trybie tylko do odczytu. Wypisanie wartości atrybutów jest możliwe w każdym przypadku. Program *tunefs* należy wówczas uruchomić z opcją **-p** podając jako argument nazwę pliku urządzenia, na którym znajduje się system plików. Charakterystyka przykładowego systemu plików jest następująca:

```

1  root@messy ~ # tunefs -p /dev/ad1s1d
2  tunefs: POSIX.1e ACLs: (-a)                      disabled
3  tunefs: NFSv4 ACLs: (-N)                          disabled
4  tunefs: MAC multilabel: (-l)                     disabled
5  tunefs: soft updates: (-n)                       disabled
6  tunefs: gjournal: (-J)                           disabled
7  tunefs: maximum blocks per file in a cylinder group: (-e) 2048
8  tunefs: average file size: (-f)                  16384

```

```

9  tuneefs: average number of files in a directory: (-s)      64
10 tuneefs: minimum percentage of free space: (-m)           8%
11 tuneefs: optimization preference: (-o)                    time
12 tuneefs: volume label: (-L)

```

Jak widać system plików nie obsługuje list kontroli dostępu (linie 2 i 3). Nie posiada również możliwości etykietowania MAC (linia 4). Istotne są braki w dziennikowaniu. Nie jest obsługiwana ani jego prostsza forma w postaci miękkich poprawek (linia 5), jak również dziennikowanie obsługiwane przez system *gjournal* (linia 6). Maksymalna liczba bloków danych przeznaczonych dla jednego pliku w obrębie jednej grupy cylindrów wynosi 2048 (linia 7). Przewidywany, średni rozmiar pliku w założonym systemie plików wynosi 16384 B (linia 8). Średnia, przewidywana liczba plików w katalogu to 64 (linia 9). Minimalny obszar systemu plików, który nie będzie dostępny dla zwykłych użytkowników to 8% całej jego pojemności (linia 10), co implikuje wykorzystanie szybkiego algorytmu alokacji bloków danych (linia 11). Systemowi plików nie została nadana żadna etykieta (linia 12). W nawiasach, po opisie danego atrybutu, została podana opcja programu *tuneefs* służąca do zmiany jego wartości. Jak widać nie jest możliwa zmiana rozmaru bloku danych oraz bloku fizycznego.

Z interesujących, a nie ustawionych jako domyślne w momencie zakładania systemu plików jest możliwość wykorzystania list kontroli dostępu dla określania praw dostępu do plików tworzonych w systemie plików. Przypomnijmy, że oprócz ustawienia tej flagi w systemie plików konieczne jest jego zamontowanie z odpowiednią opcją. Spróbujemy również zmniejszyć obszar zarezerwowany do 5%. Wykonanie programu *tuneefs* przebiegło w następujący sposób:

```

1  root@messy ~ # tuneefs -a enable -m5 /dev/ad1s1d
2  tuneefs: POSIX.1e ACLs set
3  tuneefs: minimum percentage of free space changes from 8% to 5%
4  tuneefs: should optimize for space with minfree < 8%

```

Pozwoliło to na włączenie list kontroli dostępu zgodnych z normą POSIX.1e (linia 2). Został zmniejszony obszar zarezerwowany do 5% pojemności systemu plików, a zwolnione miejsce wykorzystane przez bloki danych (linia 3). Zmniejszenie rozmiaru obszaru zarezerwowanego spowodowało automatyczną zmianę alorytmu alokacji bloków danych na optymalizowany pod kątem minimalizacji defragmentacji (linia 4). Nic nie stoi na przeszkodzie, aby powrócić do algorytmu szybkiej alokacji.

Do podstawowych opcji programu *tuneefs* należy zaliczyć:

- **-A** – system plików posiada kilka kopii superbloków. Użycie opcji powoduje uzgodnienie zawartości wszystkich superbloków na podstawie superbloku pierwszego (głównego). Opcje należy stosować z rozwagą, gdyż może się okazać, że zawartość pierwszego superbloku jest uszkodzona.
- **-a enable/disable** – włącza, wyłącza obsługę list kontroli dostępu zgodnych z POSIX1.e.
- **-e liczba\_bloków** – wartością opcji jest maksymalna liczba bloków danych, jaką zajmuje plik, aby zostały one zaalokowane jeszcze w jednej grupie cylindrów.
- **-f średni\_rozmiar\_pliku** – opcja pozwala określić średni, spodziewany rozmiar pliku w systemie plików. Wartość tego atrybutu ma wpływ na działanie algorytmu alokacji bloków danych.

- **-J enable/disable** – włącza lub wyłącza obsługę dziennikowania przez *gjournal*.
- **-L etykieta\_systemu\_plików** – opcja umożliwia nadanie nowej etykiety lub modyfikację istniejącej.
- **-l enable/disable** – opcja pozwala włączyć lub wyłączyć obsługę etykietowania dla praw dostępu MAC.
- **-m minimalny\_obszar\_zarezerwowany** – wartością opcji jest wyrażony w procentach rozmiar zarezerwowany, który nie może zostać wykorzystany przez zwykłego użytkownika. Wartość domyślna wynosi 8%. Ustawienie wartości mniejszej skutkuje zmniejszeniem wydajności systemu plików, gdyż algorytm alokacji bloków danych działa w stronę znajdowania bloków znajdujących się blisko siebie oraz współdzielenia bloków przez zawartość więcej niż jednego pliku.
- **-N enable/disable** – opcja pozwala włączyć lub wyłączyć obsługę przez system plików list kontroli dostępu zgodnych z systemem plików NFS w wersji 4.
- **-n enable/disable** – opcja umożliwia włączenie lub wyłączenie mechanizmu miękkich poprawek.
- **-o space/time** – opcja pozwala określić sposób działania algorytmu alokacji bloków danych dla pliku. Wartość *space* pozwala na zmniejszenie defragmentacji oraz współdzielenie bloków. Wydłuża jednak czas działania algorytmu. Wartość *time* pozwala na przyspieszenie działania algorytmu kosztem zwiększenia defragmentacji i zajętości bloków danych.
- **-p** – użycie opcji wypisuje jedynie informację o wartościach podstawowych atrybutów systemu plików. Szczegółowych informacji dostarcza polecenie *dumpfs*.
- **-s liczba\_plików\_na\_katalog** – wartością opcji jest spodziewana, średnia liczba plików w katalogu systemu plików.

### Podstawowe programy narzędziowe

Podstawowymi programami narzędziowymi dla systemów plików w systemie FreeBSD są programy *fsck*, *fsck\_ffs* (*fsck\_ufs*) oraz *fsdb*.

Program *fsck* weryfikuje i uspoźnia (*preens*) system plików. Podczas inicjalizacji systemu operacyjnego jest uruchamiany ze skryptu */etc/rc*. Czyta zawartość tablicy systemów plików (*/etc/fstab*) dla ustalenia, które systemy plików mają zostać poddane weryfikacji. Takie działanie ma miejsce również, gdy program zostanie uruchomiony z linii poleceń bez podania argumentu jakim ma być nazwa pliku urządzenia, na którym znajduje się analizowany system plików. Sprawdzeniu będą podlegać systemy montowane z opcjami *rw*, *rq* lub *ro*, jeśli w kolumnie opisującej kolejność weryfikacji posiadają wartość różną od zera. Systemy oznaczone wartością 1 (powinien to być główny system plików) są zawsze weryfikowane sekwencyjnie. Zasadniczo sposób działania programu i kolejność weryfikacji systemu plików zależą od sposobu jego uruchomienia i wartości kolumny opisującej kolejność weryfikacji. W trybie weryfikacji, po zakończeniu procesu uspoźniania dla systemów plików oznaczonych przez 1, program wykonuje działania dla kolejnych systemów plików według rosnącej wartości oznaczającej numer przebiegu. Systemy plików oznaczone tą samą wartością będą sprawdzane równolegle. Praca w trybie różnym od trybu weryfikacji polega na analizie kolejnych systemów plików w kolejności rosnącej wartości opisującej numer przebiegu oraz w kolejności występowania w tablicy systemów plików dla tych samych wartości w sposób sekwencyjny.

Program uruchomiony podczas inicjalizacji systemu operacyjnego faktycznie działa w dwóch etapach. Pierwszy ma miejsce zanim systemy plików zostaną zamontowane. Program zostaje uruchomiony w tle, z opcją **-F** dla sprawdzenia, czy z danego systemu plików jest inicjalizowany system operacyjny lub, czy proces weryfikacji systemu plików będzie mógł zostać przeprowadzony po jego zamontowaniu. Po zamontowaniu systemów plików weryfikowane są te, których weryfikacja jest możliwa. Program jest uruchamiany w tle z opcją **-B**. Uruchomienie programu z linii komend bez podania jako argumentu nazwy pliku urządzenia skutkuje sekwencyjnym sprawdzeniem zdefiniowanych w tablicy systemów plików, w kolejności rosnącej wartości numeru przebiegu.

Program *fsck* może zostać uruchomiony z opcjami. Do podstawowych należą:

- **-C** – opcja powoduje przeprowadzenie weryfikacji, jeśli flaga czystości (*clean*) w superbloku systemu plików jest ustawiona oraz pominięcie weryfikacji, jeśli system plików został odmontowany poprawnie i jest zaznaczony jako czysty.
- **-d** – wymusza pracę w trybie *debug* polegającym na wypisywaniu komend bez ich wykonywania. Opcja jest dostępna jeśli polecenie *fsck* zostało odpowiednio skompilowane.
- **-f** – opcja pozwala wymusić weryfikację systemu plików jeśli jest on zaznaczony jako czysty.
- **-n** – umożliwia automatyczne udzielanie negatywnych odpowiedzi na pytania zadawane przez program za wyjątkiem pytania o kontynuowanie pracy.
- **-p** – uruchamia program w trybie *preen*, w którym zostanie podjęta próba naprawy tylko wąskiej grupy niegroźnych dla systemu plików niespójności. Wykrycie błędów sprzętowych, jak np. uszkodzenie sektora dysku lub innych wywołanych błędami oprogramowania spowoduje zakończenie pracy programu *fsck*.
- **-F** – opcja pozwala uruchomić program jako zadanie pierwszoplanowe. Uruchomiony dla każdego systemu plików z tą opcją pozwala stwierdzić, czy program weryfikacyjny zostanie uruchomiony w trakcie inicjalizacji systemu operacyjnego, czy jako zadanie uruchomione w tle w działającym już systemie operacyjnym. Informację o tym niesie wartość zwrócona przez program *fsck*. Różna od zera oznacza, że wymagane jest uruchomienie zadania jako pierwszoplanowego i wywoływany jest program weryfikacyjny. Wartość równa zero powoduje uruchomienie programu weryfikacyjnego po zakończeniu inicjalizacji systemu operacyjnego, jako zadania pracującego w tle, o czym informuje stosowny komunikat.
- **-B** – użycie opcji spowoduje uruchomienie programu w tle. Program sprawdzający, dla każdego systemu plików jest uruchamiany z opcją **-F** dla stwierdzenia, czy będzie konieczne jego uruchomienie w trakcie inicjalizacji systemu operacyjnego, czy po jego inicjalizacji, jako zadania pracującego w tle. Zakończenie wykonywania programu z wartością kodu różną od zera wskazuje na konieczność uruchomienia programu weryfikującego jako zadania pierwszoplanowego, co skutkuje pominięciem montowania bieżącego systemu plików do momentu zakończenia pracy programu. Wartość zerowa kodu powrotu informuje o konieczności uruchomienia programu w tle. Działanie programu dla aktywnych systemów plików musi zostać zakończone. Uruchomienie programu w tle skutkuje sprawdzaniem sekwencyjnym systemów plików.
- **-t lista\_typów\_systemu\_plików** – wartością opcji jest lista typów systemów plików (nazwy typów oddzielone przecinkiem), które mają być sprawdzone programem *fsck*. Jeśli przed listą pojawi się słowo **no**, to sprawdzone zostaną systemy plików wszystkich typów za wyjątkiem wyspecyfikowanych w liście.

- **-v** – opcja przłącza działanie komendy w tryb gadatliwy, co w praktyce powoduje wypisanie każdego polecenia przed jego wykonaniem.
- **-y** – użycie opcji powoduje automatyczne udzielanie twierdzącej odpowiedzi na każde zadane pytanie.
- **-T typ\_systemu\_plików:opcje** – wartością opcji jest lista składająca się z oddzielonych przecinkami specyfikatorów typów systemów plików i specyficznych opcji programu oddzielonych przecinkami (jak w przypadku opcji montowania programu *mount*).

Program *fsck\_ffs* (*fsck\_ufs*) wymaga podania jako argumentu nazwy pliku urządzenia, na którym znajduje się sprawdzany system plików. Praca w trybach „*preen*” oraz „*check clean*” sprawdzane jest ustawienie flagi czystości w każdym superbloku systemu plików, a jego weryfikacja następuje jeśli flaga ta jest ustawiona. Jak wspomniano system plików jest zaznaczany jako czysty, jeśli został odmontowany, jest zamontowany w trybie tylko do odczytu lub jeśli praca programu *fsck\_ffs* na nim zakończyła się powodzeniem.

Dla jądra systemu operacyjnego szczególnie istotne jest pewna klasa niespójności występujących w systemie plików. Należą do nich:

- Osierococone i-węzły.
- Niepoprawne wartości liczników dowiązań w i-węzłach.
- Zgubione bloki z mapy bitowej bloków.
- Bloki znajdujące się w mapie bloków wolnych i jednocześnie przydzielone plikom.
- Niepoprawne wartości liczników w superbloku.

Niespójności te mogą zostać usunięte przez uruchomienie programu z opcją **-p**. Jeśli jednak w systemie plików stwierdzone zostaną inne błędy, to program zakończy działanie i zwróci wartość różną od zera. Usunięcie każdej niespójności jest raportowane przez podanie nazwy systemu plików oraz jego krótkiej charakterystyki. Usunięcie wszystkich znalezionych niespójności, wymienionych powyżej, skutkuje wypisaniem informacji o liczbie plików przechowywanych w danym systemie plików, liczbie zajętych i wolnych bloków danych oraz stopniu fragmentacji systemu plików.

Pominięcie opcji **-p** powoduje, że program we wskazanym systemie plików, w sposób interaktywny naprawia znalezione niespójności. Ponieważ naprawa niektórych z nich może prowadzić do utraty danych, program, przed każdą czynnością wymaga akceptacji (twierdząca lub przecząca odpowiedź na pytanie). Jest to domyślny sposób pracy. Jeśli użytkownik, który dokonuje sprawdzenia spójności systemu plików nie ma prawa zapisu do systemu plików, to wymagane jest uruchomienie programu z opcją **-n**.

Program *fsck\_ufs* dokonuje sprawdzenia niespójności wynikających z:

1. Numerów bloków danych, które mogą być wskazywane przez tablicę alokacji znajdującą się w więcej niż jednym i-węźle lub w co najmniej jednym i-węźle i przez listę wolnych bloków danych.
2. Numeru bloku danych zapisanego w tablicy alokacji i-węzła większego od liczby dostępnych w danym systemie plików bloków danych.
3. Niepoprawnej wartości licznika dowiązań.



4. Niepoprawnego rozmiaru katalogu (nie będącego wielokrotnością rozmiaru bloku - wartość atrybutu DIRBLKSIZ) oraz rozmiaru pliku innego niż zapisany w opisującym go i-węźle i wynikającym z tablicy alokacji.
5. Niepoprawnego formatu danych i-węźła.
6. Informacji zapisanej w katalogu:
  - numeru i-węźła wskazującego na nieistniejący plik,
  - numeru i-węźła większego od liczby i-węźłów znajdujących się w danym systemie plików,
  - istnienia w katalogu nie zapisanych bloków danych (tzw. występowanie dziur),
  - nie występowania katalogu bieżącego (.) lub katalogu nadrzędnego (..) jako odpowiednio dwóch pierwszych wpisów w katalogu lub posiadających niepoprawny numer i-węźła.
7. Informacji zapisane w bloku identyfikacyjnym:
  - większej liczby bloków przypadających na i-węzeł niż całkowita liczba bloków w systemie plików.
  - niepoprawnego formatu listy wolnych i-węźłów.
  - niepoprawnej wartości liczników wolnych bloków danych i/lub i-węźłów.

Osierocone bloki danych, należące do plików i katalogów trafiają do katalogu *lost+found*, znajdującego się w głównym katalogu systemu plików. Nazwy plików, do których są dołączane są numerami i-węźłów je opisującymi.

Jeśli do procesu *fsck\_ufs* zostanie wysłany sygnał QUIT, to zakończy on swoje działania z kodem powrotu różnym od zera (wskazującym na błąd). Jeśli miało to miejsce podczas inicjalizacji systemu operacyjnego, to zostanie ona przerwana. Jest to wykorzystywane do przerywania weryfikacji systemu plików podczas procesu automatycznego restartu systemu jeśli chcemy aby nie został on uruchomiony w trybie wieloużytkownikowym. Sygnał INFO powoduje wypisanie na standardowym wyjściu linii informującej o nazwie pliku urządzenia, dla którego dokonywane jest sprawdzanie spójności, numeru aktualnie wykonywanego przebiegu i charakterystycznych dla niego informacji.

Do podstawowych opcji programu należą:

- **-F** – uruchomienie programu z tą opcją stosuje się, gdy zachodzi konieczność ustalenia, czy usunięcie niespójności systemu plików ma zostać przeprowadzone natychmiast przez proces uruchomiony jako pierwszoplanowy, czy może zostać odroczone i przeprowadzone przez proces uruchomiony w tle. W tym przypadku wymagane jest aby system plików miał włączony mechanizm miękkich poprawek, nie był zaznaczony jako wymagający usuwania niespójności przez proces pierwszoplanowy, być zamontowanym, a w chwili kończenia pracy programu dostępnym do zapisu. Spełnienie powyższych warunków skutkuje pomyślnym zakończeniem pracy programu (kod powrotu wynosi 0). W przeciwnym przypadku program kończy pracę z kodem powrotu różnym od zera. Jeśli system plików nie posiada niespójności, uruchomiony z tą opcją program *fsck\_ufs* kończy pracę z kodem powrotu różnym od zera, co powoduje iż jego weryfikacja będzie mogła zostać przeprowadzona przez proces uruchomiony jako pierwszoplanowy. Istotnym jest, że uruchomienie z tą opcją nie usuwa żadnych niespójności, a jedynie decyduje czy mają zostać one usunięte przez proces uruchomiony w tle, czy pierwszoplanowy.

- **-B** – użycie opcji powoduje przeprowadzenie weryfikacji wskazanego argumentem, aktywnego systemu plików. Liczba wprowadzanych poprawek jest jednak ograniczona do tych, które weryfikowane są w trybie „czyszczenia” uruchamianego opcją **-p**. Napotkanie niespójności spoza zdefiniowanego zbioru skutkuje zaznaczeniem systemu plików jako wymagającego weryfikacji programem *fsck\_ffs* uruchomionym jako zadanie pierwszoplanowe oraz zakończeniem pracy programu.
- **-b numer\_superbloku** – wartością opcji jest numer bloku danych zawierający kopię superbloku. Jest ona wykorzystywana jeśli pierwszy superblok systemu plików uległ uszkodzeniu.
- **-C** – opcja jest wykorzystywana do sprawdzania spójności systemu plików po jego odmontowaniu. Jeśli przebiegło ono poprawnie, system plików nie jest sprawdzany. Jeśli system plików nie został poprawnie odmontowany, wykonywane jest pełne sprawdzenie jego poprawności.
- **-c** – pozwala przekonwertować do wskazanego poziomu pracy systemu plików. Poziom pracy może w trakcie działania systemu plików zmieniać się. Wyróżnia się cztery poziomy pracy:
  0. system plików pracuje w tzw. „starym” formacie wykorzystując tablice statyczne.
  1. system plików wykorzystuje tablice dynamiczne („nowy” format).
  2. na tym poziomie identyfikatory właściciela indywidualnego i grupowego obiektu są przechowywane z wykorzystaniem 32 bitów, krótkie dowiązania symboliczne przechowywane w i-węzłach, wpisy w katalogach posiadają dodatkowe pole informujące o typie przechowywanego w nim obiektu.
  3. jeśli wartość atrybutu *maxcontig* jest większa od jeden, w systemie plików tworzone są mapy segmentów dla szybszego wyszukiwania zbiorów bloków danych położonych obok siebie. Jeśli wartość wynosi jeden, to przejście na ten poziom pracy skutkuje usunięciem istniejących map segmentów.

Program uruchomiony w sposób interaktywny będzie informował o wykonywanych konwersjach formatu i wymagał ich akceptacji. W przypadku odpowiedzi przeczącej, żadne dalsze operacje na systemie plików nie będą wykonywane. W przypadku użycia opcji **-p**, program wypisze informację o wykonywanej konwersji lecz nie będzie wymagał jej akceptacji. Bieżący format systemu plików można ustalić programem *dumpfs*.

- **-f** – wymusza sprawdzenie spójności systemu plików, nawet jeśli jest on zaznaczony jako czysty (nie jest ustawiona flaga *clean*).
- **-m** – opcja pozwala na ustawienie praw dostępu do katalogu *lost+found*. Specyfikuje się je w formacie oktalnym. Wartość domyślna to 1777.
- **-n** – użycie opcji powoduje udzielanie odpowiedzi przeczącej na zadawane przez program pytanie za wyjątkiem pytania o kontynuowanie działania.
- **-p** – opcja uruchamia program w trybie usuwania wybranych rodzajów niespójności.
- **-y** – użycie opcji automatycznie udziela odpowiedzi twierdzącej (yes) na każde pytanie zadawane przez program.

Działanie programów narzędziowych prześledzimy na przykładzie. W systemie zainstalowany został dodatkowy dysk o pojemności 80 GB, reprezentowany plikiem */dev/ad15*. Został on podzielony na dwa paski o zbliżonych rozmiarach. Na pierwszym pasku została założona partycja o rozmiarze ok. 19 GB, na której założono system plików ufs z domyślnymi wartościami atrybutów:

```

1 [root@messy ~]# newfs /dev/ad15s1d
2 /dev/ad15s1d: 19000.0MB (38912000 sectors) block size 16384, fragment size 2048
3     using 104 cylinder groups of 183.77MB, 11761 blks, 23552 inodes.
4 super-block backups (for fsck -b #) at:
5     160, 376512, 752864, 1129216, 1505568, 1881920, 2258272, 2634624, 3010976,
6     ....

```

System plików został następnie zamontowany w katalogu */scratch1* i udostępniony do eksploatacji. Pojawiły się w nim dwa katalogi z zawartością. Uruchomienie programu *fsck\_ufs* dało następujący wynik:

```

1 [root@messy ~]# fsck_ufs /dev/ad15s1d
2 ** /dev/ad15s1d (NO WRITE)
3 ** Last Mounted on /scratch1
4 ** Phase 1 - Check Blocks and Sizes
5 ** Phase 2 - Check Pathnames
6 ** Phase 3 - Check Connectivity
7 ** Phase 4 - Check Reference Counts
8 ** Phase 5 - Check Cyl groups
9 2687 files, 32422 used, 9387697 free (33 frags, 1173458 blocks, 0.0% fragmentation)

```

System plików został uszkodzony przez skopiowanie na jego początek 15 bloków o rozmiarze 64 kB i przypadkowej wartości:

```

1 [root@messy ~]# dd if=/dev/urandom of=/dev/ad15s1d bs=65536 count=15
2 15+0 records in
3 15+0 records out
4 983040 bytes transferred in 0.028341 secs (34686077 bytes/sec)

```

Zamontowanie uszkodzonego systemu plików nie powiodło się:

```

1 [root@messy ~]# mount /dev/ad15s1d /scratch1
2 mount: /dev/ad15s1d : Invalid argument

```

Uruchomienie programu *fsck\_ufs* również zakończyło się niepowodzeniem, co pokazują linie 1–5 poniższego listingu. Powodem było uszkodzenie podstawowego bloku identyfikacyjnego. Stąd drugie uruchomienie z opcją **-b** pozwalającą wyspecyfikować numer cylindra, na którym znajduje się kopia superbloku (linia 6):

```

1 [root@messy ~]# fsck_ufs /dev/ad15s1d
2 ** /dev/ad15s1d
3 Cannot find file system superblock
4 ioctl (GCINFO): Inappropriate ioctl for device
5 fsck_ufs: /dev/ad15s1d: can't read disk label
6 [root@messy ~]# fsck_ufs -b 376512 /dev/ad15s1d
7 Alternate super block location: 376512
8 ** /dev/ad15s1d

```

```

 9  ** Last Mounted on
10  ** Phase 1 - Check Blocks and Sizes
11  CYLINDER GROUP 0: BAD MAGIC NUMBER
12  REBUILD CYLINDER GROUP? [yn] y
13
14  ** Phase 2 - Check Pathnames
15  ROOT INODE UNALLOCATED
16  ALLOCATE? [yn] y
17
18  ...
19  ** Phase 3 - Check Connectivity
20  UNREF DIR I=612352 OWNER=root MODE=40755
21  SIZE=5120 MTIME=Mar 9 13:20 2011
22  RECONNECT? [yn] y
23
24  NO lost+found DIRECTORY
25  CREATE? [yn] y
26
27  ....
28  ** Phase 4 - Check Reference Counts
29  LINK COUNT DIR I=376832 OWNER=root MODE=40755
30  SIZE=11776 MTIME=Mar 9 13:21 2011 COUNT 6 SHOULD BE 5
31  ADJUST? [yn] y
32
33  ** Phase 5 - Check Cyl groups
34  FREE BLK COUNT(S) WRONG IN SUPERBLK
35  SALVAGE? [yn] y
36
37  SUMMARY INFORMATION BAD
38  SALVAGE? [yn] y
39
40  BLK(S) MISSING IN BIT MAPS
41  SALVAGE? [yn] y
42
43  2686 files, 32421 used, 9387698 free (18 frags, 1173460 blocks, 0.0% fragmentation)
44
45  UPDATE STANDARD SUPERBLOCK? [yn] y
46
47  ***** FILE SYSTEM IS CLEAN *****
48  ***** FILE SYSTEM WAS MODIFIED *****

```

Ponowne montowanie zakończyło się sukcesem. Jednak zawartość niektórych i-węzłów jest uszkodzona, co pokazuje poniższy listing:

```

1  [root@messy ~]# mount /dev/ad15s1d /scratch1
2  [root@messy ~]# ls -ldi /scratch1
3  2 drwxr-xr-x 2 4246250838 837066117 2048 Mar 9 13:57 /scratch1

```

Zmianę właścicieli głównego katalogu naszego, eksperymentalnego systemu plików możemy

dokonać korzystając z programu zarządzającego *fsdb*, pamiętając aby przed jego uruchomieniem odmontować system plików. Bieżącym i-węzłem edytowanym przez program *fsdb*, jest i-węzeł o numerze 2 opisujący główny katalog systemu plików. Dokonujemy zmiany właścicieli indywidualnego i grupowego, co przedstawiono poniżej:

```

1  [root@messy ~]# fsdb /dev/ad15s1d
2  ** /dev/ad15s1d
3  Editing file system '/dev/ad15s1d'
4  Last Mounted on /scratch1
5  current inode: directory
6  I=2 MODE=40755 SIZE=2048
7      BTIME=May 18 06:43:03 2024 [-706919144 nsec]
8      MTIME=Mar  9 13:57:54 2011 [0 nsec]
9      CTIME=Mar  9 13:57:54 2011 [0 nsec]
10     ATIME=Mar  9 14:15:47 2011 [0 nsec]
11  OWNUID=4246250838 GID=837066117 LINKCNT=2 FLAGS=0 BLKCNT=4 GEN=6319eb56
12  fsdb (inum: 2)> chown root
13  fsdb (inum: 2)> chgrp wheel
14  current inode: directory
15  I=2 MODE=40755 SIZE=2048
16      BTIME=May 18 06:43:03 2024 [-706919144 nsec]
17      MTIME=Mar  9 13:57:54 2011 [0 nsec]
18      CTIME=Mar  9 13:57:54 2011 [0 nsec]
19      ATIME=Mar  9 14:15:47 2011 [0 nsec]
20  OWNER=root GRP=wheel LINKCNT=2 FLAGS=0 BLKCNT=4 GEN=6319eb56
21  fsdb: rval was 1
22  fsdb (inum: 2)> q
23
24  ***** FILE SYSTEM STILL DIRTY *****
25  *** FILE SYSTEM MARKED DIRTY
26  *** BE SURE TO RUN FSCK TO CLEAN UP ANY DAMAGE
27  *** IF IT WAS MOUNTED, RE-MOUNT WITH -u -o reload

```

System plików jest zaznaczony jako brudny. Stąd konieczne jest ponowne uruchomienie polecenia *fsck\_ufs*. Pozwala ono uzyskać informacje o kolejnych błędach w systemie plików i usunąć je programem *fsdb*. Iteracje te powtarzamy, aż uszkodzony system plików zostanie doprowadzony do stanu spójnego.

### 3.5.4 System plików ZFS (Z file system)

Firma SUN rozpoczęła prace nad systemem plików ZFS w roku 2001. Pierwsze informacje na jego temat pojawiły się w roku 2004. Integracja z systemami operacyjnymi Solaris oraz OpenSolaris została rozpoczęta w roku 2005. W roku 2006 ZFS pojawił się w wersji 10 systemu Solaris, a rok później w systemie OpenSolaris. ZFS stanowi połączenie systemu plików z zarządzcą przestrzeni logicznej (LVM). Takie rozwiązanie pozwoliło na stworzenie programowych macierzy dyskowych RAID-Z. System ten posiada dodatkowo wiele zaawansowanych rozwiązań, do których należy zaliczyć zabezpieczenia przed fałszowaniem danych, możliwość obsługiwanie urządzeń o dużych pojemnościach, tworzenie kopii migawkowych oraz klonowanie przy zapisie. Dla osiągnięcia dużych wydajności przy jednoczesnym zapewnieniu małego obciążenia systemu operacyjnego operacjami wejścia/wyjścia zaimplementowano w nim mechanizm zarządzania pamięcią podręczną

na wielu poziomach. Ogólnemu zwiększeniu wydajności systemu zbudowanemu w oparciu o wielu urządzeń, tworzących odpowiednik macierzy dyskowej zwany w nomenklaturze ZFS *zpool* służy mechanizm dynamicznego paskowania (*Dynamic striping*). Jak każdy nowoczesny system plików, ZFS został wyposażony w mechanizm list kontroli dostępu. ZFS jest systemem 128-bitowym. Jego nazwa, pochodząca od *Zettabyte File System*, wskazuje na możliwości przechowywania ogromnych ilości danych, jak i na możliwości dostępu do nich w akceptowalnym czasie. Szacuje się, że wypełnienie danymi systemu plików ZFS o jego maksymalnym fizycznym rozmiarze będzie wymagało więcej energii niż koniecznej do zagotowania całej wody na Ziemi. Stąd w przypadku nowoczesnych systemów plików już nie mówi się o ich ograniczeniach fizycznych.

System plików ZFS jest implementowany jako oprogramowanie o kodzie otwartym, licencjonowanym na zasadach zgodnych z CDDL (*Common Development and Distribution Licence*). Pewne niezgodności pomiędzy CDDL, a GNU GPL (*General Public License*) powodują iż nie może on zaistnieć legalnie w systemach linuksowych. Jest jednak dostępny w wielu innych, jak np. rodzinie BSD czy Mac OS X. Znak handlowy ZFS jest od 20 września 2011 roku własnością firmy Oracle.

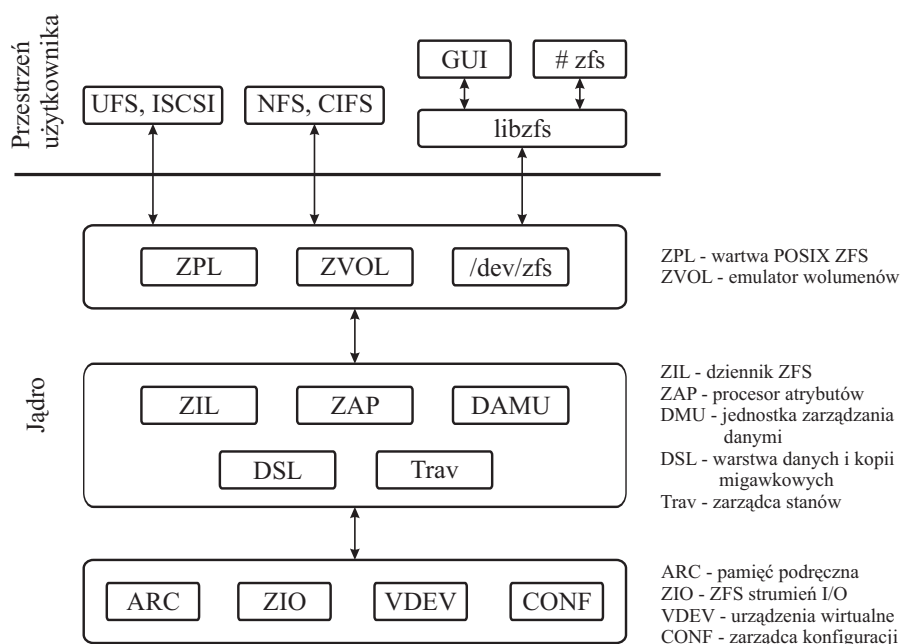
### Struktura wewnętrzna

Ogólnie, w logicznej strukturze ZFS, pracującej w warstwie jądra systemu operacyjnego, wyróżnia się trzy warstwy (Rysunek 3.37). Warstwa najniższa zwana jest warstwą *zpool*–administracji. Komunikuje się ona bezpośrednio ze sprzętem wykorzystując jego sterowniki. Implementacja operacji niskopoziomowych w kodzie systemu operacyjnego pozwala na zarządzanie przestrzenią logiczną bez konieczności stosowania zewnętrznego zarządcy w postaci chociażby systemu LVM. Warstwa druga jest odpowiedzialna za przeprowadzanie transakcji oraz operacji związanych z sumami kontrolnymi, które stanowią istotną część niniejszego systemu plików, służącą wykrywaniu fałszowania danych spowodowanego zarówno przez niskopoziomowe oprogramowanie dysków jak i przez czynniki zewnętrzne. Podstawowe mechanizmy działania tej warstwy oparte są o tzw. drzewa Merkle (*Merkle-Trees*), których działanie jest kontrolowane właśnie w tej warstwie. Warstwa najwyższa służy komunikowaniu się z przestrzenią użytkownika. Dostarcza ona również interfejsów do innych usług, umożliwiających dostęp do sprzętowych interfejsów urządzeń, na których założono system plików ZFS.

Istotnym pojęciem, które pojawiło się w systemie plików ZFS jest pojęcie puli (*pool*). Pojawia się ono przede wszystkim w kontekście łączenia urządzeń fizycznych (*physical storage devices*) dla uzyskania ciągłej, dyskowej przestrzeni adresowej (*logical unit*). Tak rozumianą pulę tworzą partycje logiczne, których liczba jest praktycznie dowolna. Różnica w stosunku do klasycznego rozwiązania z wykorzystaniem LVM polega na tym, że LVM dostarcza ciągłej przestrzeni dyskowej dla dowolnego systemu plików, ale tworzące ją urządzenia są jej dedykowane i nie mogą zostać przeniesione do innego wolumenu. Przeniesienie zarządzania wolumenami logicznymi do systemu plików jest rozwiązaniem bardziej elastycznym, gdyż umożliwia wykorzystanie dowolnej partycji logicznej, znajdującej się w puli, do założenia wolumenu logicznego, na którym zostanie założony system plików ZFS (Rysunek 3.37).

Pojęcie puli pojawia się również w odniesieniu do macierzy dyskowych. W takim przypadku pula oznacza zbiór urządzeń logicznych tworzących macierz dyskową. Sterowniki zaimplementowane w ZFS zapewniają bowiem funkcjonalności macierzy dyskowych. Dostępne są następujące poziomy:

- **ZFS–MIRROR** – to zbiór fizycznych dysków tworzących macierz na poziomie RAID–1. Dane na fizycznych dyskach są identyczne.
- **ZFS–RAIDZ** – rozwiązanie to jest funkcjonalnie podobne do RAID–5 i wymaga użycia

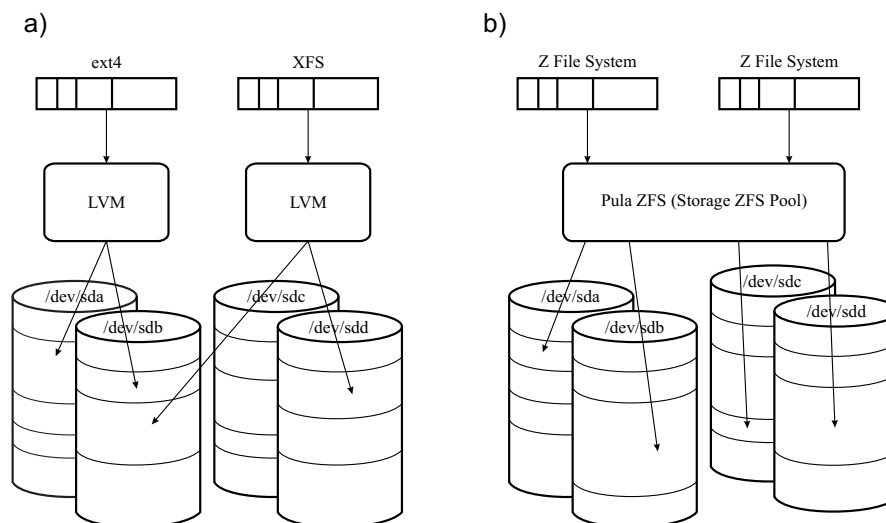


Rysunek 3.35: Struktura wewnętrzna systemu plików ZFS.

co najmniej dwóch dysków fizycznych. Bloki danych oraz bloki zawierające sumę kontrolną są zapisywane na wszystkich dyskach tworzących pulę. Założenie RAIDZ na dwóch wolumenach powoduje, iż suma kontrolna będzie identyczna z danymi. Stąd jedynie podobieństwo do RAID-5 oraz wzór określający pojemność takiej puli:  $(N - P) \times X$ , gdzie  $N$  jest liczbą wolumenów tworzących pulę,  $P$  jest liczbą dysków przeznaczonych do przechowywania sumy kontrolnej, a  $X$  pojemnością najmniejszego dysku wchodzącego w skład puli.

- **ZFS-RAIDZ1** – odpowiada rozwiązaniu RAID-5, gdyż wykorzystuje jedynie pojedynczą sumę kontrolną. W praktyce wymaga zastosowania co najmniej trzech wolumenów, gdyż dwa przechowują dane, a suma kontrolna zostaje zapisana na dysku z którego nie pochodzą dane do jej wyznaczenia.
- **ZFS-RAIDZ2** – to propozycja odpowiadająca funkcjonalnie maierzom RAID na poziomie 6. Zapewnia zatem zapis danych z podwójną, rozproszoną sumą kontrolną. Rozwiązanie to zabezpiecza przed koniecznością natychmiastowej odbudowy macierzy po awarii jednego dysku. W praktyce posiada praktyczne zastosowanie, gdyż na rynku czasami trafiają się serie wadliwych dysków. Ich wykorzystanie do budowy macierzy może, w przypadku „niższych” poziomów RAID, prowadzić do problemów. Stąd poziom ten zaleca się stosować w uzasadnionych przypadkach.

COW



Rysunek 3.36: a) Zarządzanie przestrzenią dyskową z wykorzystaniem systemu LVM. b) System plików ZFS posiada wbudowane mechanizmy zarządzania przestrzenią dyskową puli.

### Zakładanie systemu plików ZFS

### Montowanie systemu plików ZFS

### Programy narzędziowe

### 3.5.5 Systemy plików z rodziny ext w systemie FreeBSD

Jądro systemu FreeBSD od wersji 5 umożliwia obsługę systemów plików z linuksowej rodziny ext. Niezbędne oprogramowanie dystrybuowane jest w podstawowej wersji systemu. Znajduje się w katalogu `/usr/ports/sysutils/e2fsprogs`. Należy je skompilować i zainstalować. Wszystko sprowadza się do dwóch poleceń:

```
1 [root@messy ~]# cd /usr/ports/sysutils/e2fsprogs
2 [root@messy /usr/ports/sysutils/e2fsprogs]# make install all
```

### Konfiguracja jądra systemu operacyjnego

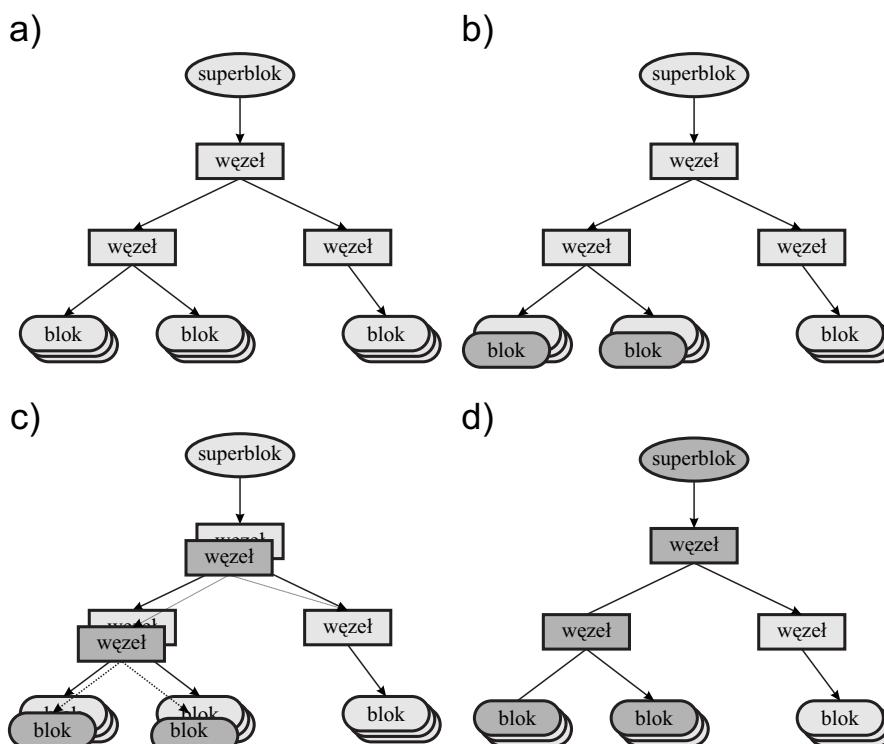
### Instalacja programów narzędziowych

### Zakładanie i montowanie systemu plików ext

## 3.6 Logical Volume Manager (LVM) w dystrybucjach RedHat

System ten powstał z myślą o możliwości tworzenia wolumenów, których rozmiar przekracza rozmiar pojedynczego dysku. Mówiąc inaczej, jego zadaniem jest „zasłanianie” ganic pomiędzy dyskami fizycznymi oraz utworzonymi na nich partycjami. W efekcie otrzymujemy wygodne narzędzie pozwalające w sposób płynny zmieniać (zwiększać lub zmniejszać) rozmiar systemu





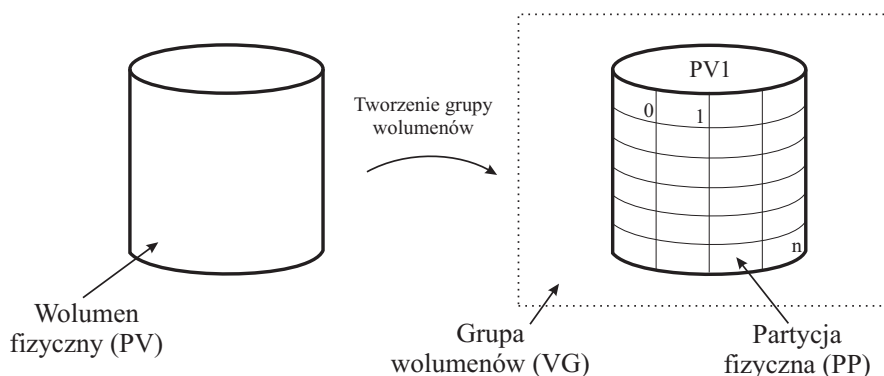
Rysunek 3.37: Sposób przeprowadzenia operacji zapisu w systemie plików ZFS wykorzystujący mechanizm „kopiowania przy zapisie”.

plików, zarówno jak chodzi o liczbę i-węzłów jak również liczbę bloków danych. Jeśli chodzi o bezpieczeństwo przechowywania danych, to w oryginalnej implementacji w systemie AIX partycja logiczna mogła być mapowana na maksymalnie trzy partycje fizyczne. W praktyce oznacza to, że dane przechowywane w systemie plików są zapisane w trzech fizycznych kopiach. W niektórych implementacjach zrezygnowano z tych możliwości.

Ponieważ pojawiły się pierwsze specyficzne dla systemu terminy, rozpoczniemy od ich zdefiniowania, a następnie omówimy proces tworzenia partycji logicznej, zakładania na niej systemu plików oraz elastycznej zmiany jego rozmiaru.

Dostępne w systemie LVM zasoby możemy podzielić na *fizyczne* i *logiczne*. Każdy zasób fizyczny, mówiąc obrazowo, możemy „wziąć do ręki”. Zasoby logiczne są tworem powstałymi przez zorganizowanie zasobów fizycznych w złożone struktury. Podstawowym zasobem fizycznym jest wolumen fizyczny (ang. *Physical Volume, PV*), który odpowiada dyskowi twardemu (rys. 3.38). Wolumeny fizyczne zamyka się w grupy wolumenów fizycznych (ang. *Volume Group, VG*). W systemie komputerowym mogą istnieć maksymalnie 32 grupy wolumenów, z których każda może zawierać do 256 wolumenów fizycznych (zatem system może teoretycznie posiadać 8192 dyski). W momencie dołączania wolumenu fizycznego do grupy dokonywany jest jego podział na jednostki alokacji zwane partycjami fizycznymi (ang. *Physical Partitions, PP*), które stanowią ciągle obszary przestrzeni adresowej dysku. Ich rozmiar jest określany przez administratora systemu i może wahać się od 8 KB do 16 GB. Rozmiar domyślny wynosi 4 MB.

Wykorzystując partycje fizyczne pochodzące z wolumenów fizycznych znajdujących się w jednej grupie wolumenów budujemy wolumen logiczny (ang. *Logical volume, LV*). Wolumen lo-



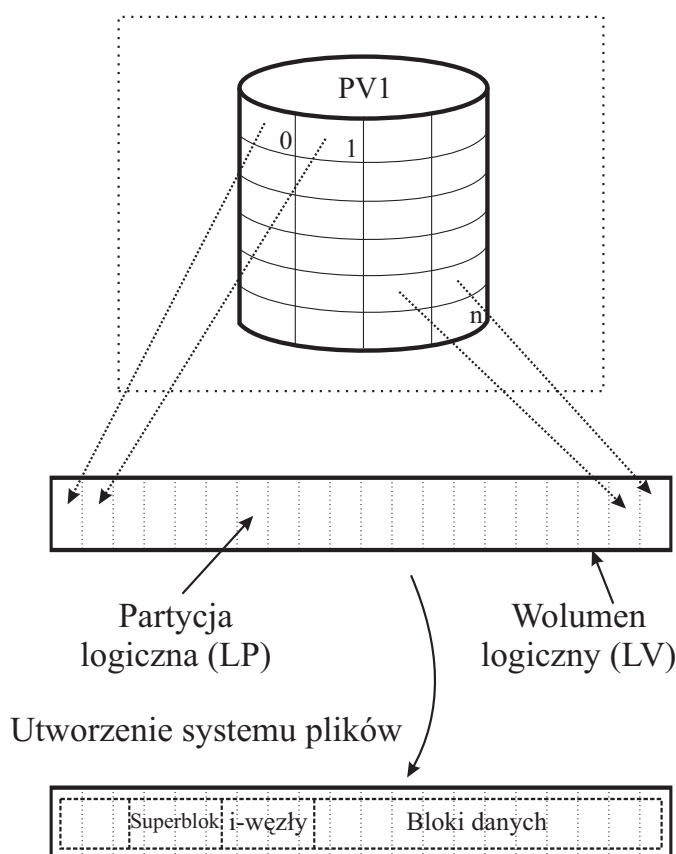
Rysunek 3.38: W oparciu o jeden wolumen fizyczny tworzymy grupę wolumenów. Dołączenie wolumenu fizycznego do grupy wymaga podzielenie go na partycje fizyczne o określonym rozmiarze. Rozmiar ten jest taki sam dla wszystkich wolumenów fizycznych w grupie.

giczny jest podzielony na fragmenty zwane partycjami logicznymi (ang. *Logical Partition, LP*), które mapowane są na partycje fizyczne. Mapowanie wymaga utworzenia w systemie odpowiedniej struktury danych odwzorowującej partycje logiczne na fizyczne. Sposób mapowania może uwzględniać zarówno liczbę partycji fizycznych, na które odwzorowane zostanie pojedyncza partycja logiczna oraz fakt czy partycje fizyczne pochodzą będą z jednego czy wielu wolumenów fizycznych, jak również, w którym obszarze wolumenu fizycznego ma znajdować się partycja fizyczna przechowująca dane odpowiedniej partycji logicznej. Stosowne polityki w oryginalnej nomenklaturze noszą nazwy odpowiednio *inter* oraz *intra* i obowiązują na poziomie wolumenu logicznego. Zostaną one omówione na zakończenie niniejszego podrozdziału. Rysunek 3.39 przedstawia sytuację, w której partycja logiczna utworzonego wolumenu logicznego mapowana jest na jedną partycję fizyczną. Kolejne partycje logiczne odpowiadają występującym po sobie na wolumenie fizycznym partycjom fizycznym.

Wolumen logiczny z punktu widzenia systemu operacyjnego jest urządzeniem blokowym, posiadającym reprezentujący go plik w katalogu `/dev`. Dla nas jest on widziany jak ciągły obszar przestrzeni adresowej dysku. Możemy zatem założyć na nim system plików, co przedstawiono na rysunku 3.39.

Utworzony system plików udostępniamy do normalnej eksploatacji. Tworzone są w nim pliki, wykorzystujące zarówno i-węzły jak i bloki danych. Generalnie dostępne zasoby systemu plików powoli kurczą się, aż w końcu ulegają wyczerpaniu. W klasycznym rozwiązaniu, bez zastosowania systemu LVM, sprawa jest problematyczna, gdyż należy wykonać kopię zapasową danych oraz przeprowadzić zmianę podziału dysku na partycje lub utworzyć partycje o większym rozmiarze. Następnie na nowo utworzonej partycji założyć system plików i na koniec skopiować dane. Operacjami najbardziej czasochłonnymi są operacje tworzenia kopii danych oraz odtwarzanie danych z kopii. Dodatkowo, podczas wykonywania opisanych system plików nie powinien być dostępny dla użytkowników. Widać, że rozwiązanie to jest kłopotliwe, a w systemach produkcyjnych wręcz niemożliwe do zastosowania.

System LVM zapewnia większą elastyczność. Jeśli zasoby systemu plików uległy wyczerpaniu, to należy sprawdzić, czy w obrębie grupy wolumenów, w której został utworzony wolumen logiczny, na którym założony został nasz system plików są jeszcze nie wykorzystane partycje fizyczne. Załóżmy, że grupa wolumenów zawiera nie wykorzystane jeszcze partycje fizyczne. Chcąc zwiększyć dostępne w systemie plików zasoby wykonujemy dwie czynności. Pierwsza polega na zwiększeniu rozmiaru partycji logicznej. W tym celu wykorzystywane są dostępne partycje fi-

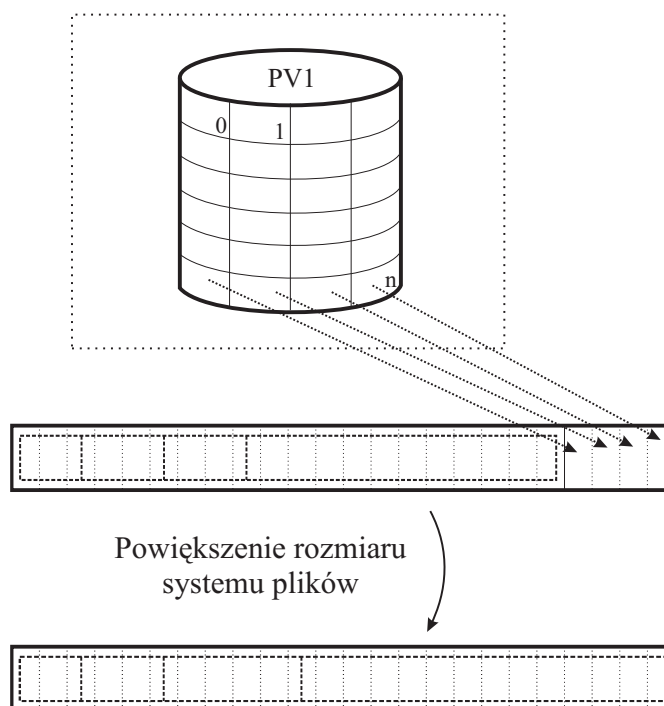


Rysunek 3.39: Operacja mapowania pozwala na zbudowanie wolumenu logicznego. Powstaje on z partycji fizycznych wolumenu. Dostępna przestrzeń adresowa jest wykorzystywana do założenia systemu plików (sformatowania).

zyczne. Faktycznie przestrzeń adresowa uległa zwiększeniu. W drugim kroku musimy „naciągnąć” istniejący system plików na obszar powiększonej partycji fizycznej. Przedstawiono to na rysunku 3.40. Należy nadmienić, iż praktycznie obie te czynności można wykonać na udostępnionym systemie plików, a użytkownicy nie zauważą prowadzonych prac. Narzędzia służące zwiększaniu rozmiaru systemu plików zwiększają zarówno liczbę dostępnych bloków danych jak również liczbę i-węzłów dostępnych w systemie plików.

Powiększony system plików pracuje do momentu wyczerpania się dostępnych w nim zasobów. Bogatsi o doświadczenia z sytuacji, która hipotetycznie miała miejsce w naszym systemie, szukamy wolnych partycji fizycznych. Ale niestety. W grupie wolumenów wykorzystaliśmy wszystkie. Niestety nie możemy wykorzystać wolnych partycji fizycznych z innej grupy wolumenów, jeśli taka istniałaby w systemie. Źródłem partycji fizycznych jest wolumen fizyczny i dlatego w zaistniałej sytuacji musimy rozszerzyć grupę wolumenów o nowy wolumen fizyczny. Mówiąc bardziej obrazowo musimy dołożyć do systemu nowy dysk i dodać go do grupy wolumenów. Zostało to przedstawione na rysunku 3.41.

W momencie dodawania nowego wolumenu fizycznego do istniejącej grupy wolumenów system dokonuje jego podziału na partycje fizyczne. W obrębie grupy wolumenów obowiązuje jeden

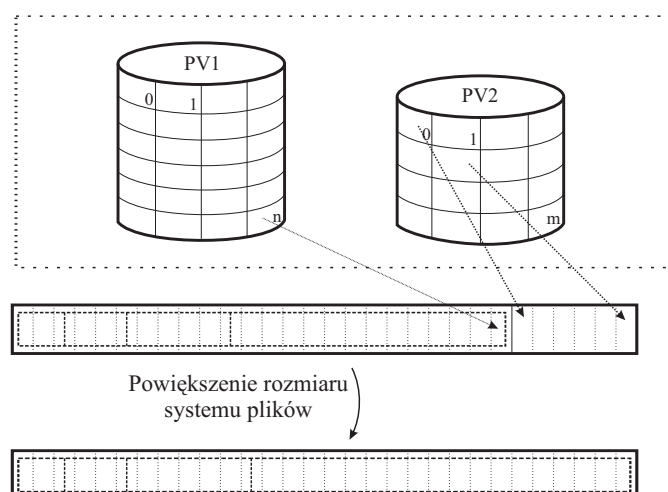


Rysunek 3.40: Mając do dyspozycji w grupie wolumenów nie wykorzystane partycje fizyczne zwiększamy rozmiar wolumenu logicznego. Następnie zwiększamy rozmiar znajdującego się na nim systemu plików.

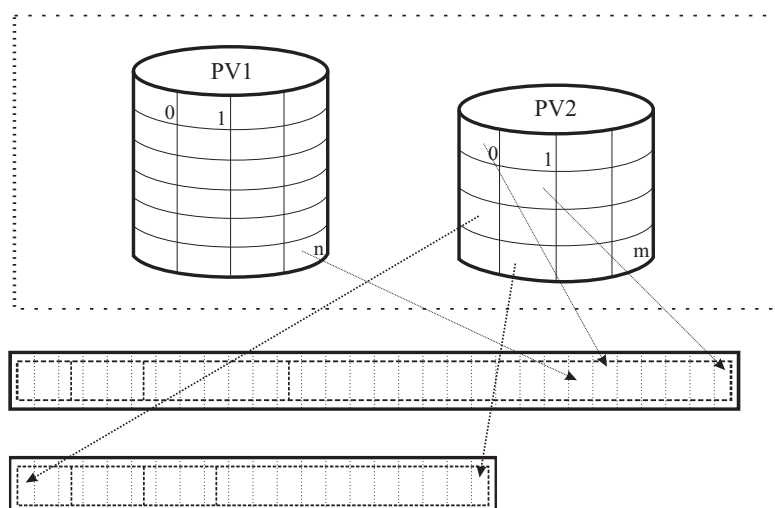
rozmiar partycji fizycznej. Czyli tworząc grupę wolumenów narzucamy rozmiar partycji fizycznej, który będzie obowiązywał dla wszystkich należących do niej wolumenów fizycznych. Partycje fizyczne znajdujące się na dodanym wolumenie fizycznym mogą być wykorzystane do zwiększenia rozmiaru każdego wolumenu logicznego utworzonego w tej grupie. W następnej kolejności zwiększamy liczbę dostępnych zasobów systemu plików. Oczywiście wszystkie wymienione czynności możemy wykonać na normalnie pracującym systemie.

Nic nie stoi na przeszkodzie, aby w obrębie grupy wolumenów zbudować kilka wolumenów logicznych. Sytuację taką przedstawiono na rysunku 3.42. Na wolumenie fizycznym PV2 pozostały jeszcze niewykorzystane partycje fizyczne. Utworzono z nich drugi wolumen logiczny, na którym założono system plików.

Jak wynika z przedstawionego powyżej krótkiego opisu, system LVM dostarcza mechanizmów pozwalających na płynne administrowanie zasobami systemów plików. Dla początkujących największym problemem jest opanowanie nomenklatury. Użyta w przedstawionych przykładach pochodzi z systemu AIX. Niestety dla zastosowań w systemach linuxowych, LVM musiał zostać zmodyfikowany. Zauważmy, że do tej pory nie zostało nic powiedziane o partycjach w rozumieniu „biosowym”, z którymi mamy do czynienia na komputerach klasy PC. Przejście jest bardzo proste. Dysk fizyczny w systemie LVM zaimplementowanym dla systemów linuxowych nazywany jest *medium fizycznym*. Na nim zakładane są partycje „biosowe”. Partycja biosowa odpowiada wolumenowi fizycznemu w nomenklaturze AIX. Jest ona podzielona na fizyczne ekstendy (ang. *Physical Extend*, PE), które odpowiadają pojęciowo partycjom fizycznym. Cała reszta jest identyczna jak w oryginalnej implementacji.



Rysunek 3.41: Do istniejącej grupy wolumenów dodajemy nowy wolumen fizyczny (PV2). Wykorzystując znajdujące się na nim partycje fizyczne do zwiększenia rozmiaru wolumenu logicznego. Następnie zwiększamy rozmiar znajdującego się na nim systemu plików.



Rysunek 3.42: Z partycji fizycznych znajdujących się na wolumenach fizycznych zamkniętych w jednej grupie wolumenów można zbudować wiele wolumenów logicznych.

Pracę z systemem LVM w dystrybucjach linuxowych rozpoczynamy od uruchomienia komendy *vgscan*. Dokonuje ona przeszukania dostępnych w systemie dysków w celu znalezienia istniejących grup wolumenów. Na marginesie, liczba występujących w systemie LVM struktur fizycznych i logicznych jest duża. Do obsługi każdej z nich jest dostępnych od kilku do kilkunastu komend. Ich nazwy zaczynają się od liter, które są pierwszymi literami słów opisujących dany zasób w języku angielskim. Przykładowo każda komenda zaczynająca się literami *vg* służy do zarządzania grupami wolumenów, a komenda zaczynająca się literami *lv* do zarządzania wolumenami logicznymi. Wracając do naszego systemu, uruchomienie komendy *vgscan* dało następujący

wynik:

```

1 [root@messy ~]# vgscan
2   Reading all physical volumes.  This may take a while...
3   No volume groups found

```

Zatem, w naszym systemie nie istnieje żadna grupa wolumenów. Tworzenie grupy wolumenów rozpoczynamy od utworzenia partycji (biosowej), która zostanie następnie przekształcona do wolumenu fizycznego. Sprawdźmy zatem, jak wygląda wykorzystanie dysków (mediów) fizycznych w naszym systemie:

```

1 [root@messy ~]# fdisk -l
2
3 Disk /dev/sda: 10.2 GB, 10202050560 bytes
4 255 heads, 63 sectors/track, 1240 cylinders
5 Units = cylinders of 16065 * 512 = 8225280 bytes
6 Disk identifier: 0xcd9acd9a
7
8      Device Boot      Start         End      Blocks   Id  System
9  /dev/sda1  *           1           65       522081   83  Linux
10 /dev/sda2             66          587      4192965   83  Linux
11 /dev/sda3           588          848      2096482+   83  Linux
12 /dev/sda4           849         1240      3148740    5  Extended
13 /dev/sda5           849         1109      2096451   83  Linux
14 /dev/sda6        1110         1174       522081   82  Linux swap / Solaris
15
16 Disk /dev/sdb: 10.2 GB, 10202050560 bytes
17 255 heads, 63 sectors/track, 1240 cylinders
18 Units = cylinders of 16065 * 512 = 8225280 bytes
19 Disk identifier: 0x90909090
20
21      Device Boot      Start         End      Blocks   Id  System
22  /dev/sdb1  *           1         1044      8385898+   83  Linux
23
24 Disk /dev/sdc: 15.3 GB, 15303075840 bytes
25 255 heads, 63 sectors/track, 1860 cylinders
26 Units = cylinders of 16065 * 512 = 8225280 bytes
27 Disk identifier: 0x000c83ce
28
29      Device Boot      Start         End      Blocks   Id  System
30

```

Widzimy, że dysk `/dev/sda` posiada 1240 cylindrów, z czego dostępne są cylindry 1175–1240, dysk `/dev/sdb` posiada 1240 cylindrów, z czego dostępne są cylindry 1045–1240, zaś dysk `/dev/sdc` posiada 1860 cylindrów i nie została na nim utworzona żadna partycja, czyli jest cały dostępny. Na dysku (medium fizycznym) `/dev/sda` tworzymy nową partycję na całej, dostępnej przestrzeni czyli cylindrach 1175–1240. W naszym przypadku otrzymuje ona numer 7 i będzie reprezentowana przez plik `/dev/sda7`. Po utworzeniu zmieniamy jej identyfikator na *Linux LVM* (wartość 8e szesnastkowo), a pracę z programem `fdisk` kończymy komendą `w`. Następnie, z wiadomych

powodów, wykonujemy komendę *partprobe*. Układ partycji na dysku */dev/sda* jest teraz następujący:

```

1 [root@messy ~]# fdisk -l /dev/sda
2
3 Disk /dev/sda: 10.2 GB, 10202050560 bytes
4 255 heads, 63 sectors/track, 1240 cylinders
5 Units = cylinders of 16065 * 512 = 8225280 bytes
6 Disk identifier: 0xcd9acd9a
7
8      Device Boot      Start         End      Blocks   Id  System
9  /dev/sda1  *           1           65       522081   83  Linux
10 /dev/sda2             66          587      4192965   83  Linux
11 /dev/sda3           588          848      2096482+   83  Linux
12 /dev/sda4           849         1240      3148740    5  Extended
13 /dev/sda5           849         1109      2096451   83  Linux
14 /dev/sda6          1110         1174       522081   82  Linux swap / Solaris
15 /dev/sda7          1175         1240      530113+   8e  Linux LVM
16

```

Kolejnym krokiem jest „przekształcenie” nowo utworzonej partycji do wolumenu fizycznego. Służy do tego komenda *pvcreeate*, która w najprostszym sposobie uruchomienia jako argumentu wymaga podania ścieżki dostępu do pliku reprezentującego w systemie utworzoną partycję:

```

1 [root@messy ~]# pvcreate /dev/sda7
2   Physical volume "/dev/sda7" successfully created

```

Po wykonaniu komendy *pvcreeate* mamy dostępny w systemie volumen fizyczny. Kolejnym krokiem jest utworzenie grupy wolumenów fizycznych. Zgodnie z przytoczoną konwencją nazwa komendy będzie zaczynać się na *vg*, a dalej pojawi się *create* od *utwórz* – stąd *vgcreate*. Komenda ta wymaga podania jako pierwszego argumentu nazwy tworzonej grupy wolumenów, zaś jako kolejne powinny pojawić się ścieżki dostępu do plików reprezentujących partycje przekształcone do wolumenów fizycznych. Spośród opcji komendy należy wskazać na opcję *-s* której wartość jest rozmiarem fizycznego ekstendu (partycji fizycznej), na które zostanie podzielony volumen fizyczny. Rozmiar może zostać podany w kilo– (kK), Mega– (mM), Giga– (gG), Peta– (pP) lub Egza– (eE) bajtach. Rozmiar domyślny wynosi 4 MB (historycznie pierwszy). W trakcie pracy grupy wolumenów rozmiar partycji fizycznej (fizycznego ekstendu) może zostać zmieniony. Służy do tego komenda *vgchange*. Utwórzmy zatem grupę wolumenów o nazwie *rootvg*, z rozmiarem fizycznego ekstendu wynoszącym 8 MB z wykorzystaniem jedyne go dostępnego w systemie urządzenia fizycznego reprezentowanego przez plik */dev/sda7*:

```

1 [root@messy ~]# vgcreate -s 8M rootvg /dev/sda7
2   Volume group "rootvg" successfully created

```

Informację o istniejących w systemie grupach wolumenów można uzyskać dzięki komendzie *vgdisplay*. Wywołana bez argumentu dostarcza informacji o wszystkich grupach wolumenów dostępnych w systemie. Jeśli jako argumet wywołania zostanie użyta nazwa występującej w systemie nazwa grupy wolumenów, to wypisana informacji będzie dotyczyć jedynie jej. Spośród opcji

komendy *vgdisplay* najczęściej wykorzystywaną jest opcja *-v*, umożliwiającą dostarczanie informacji w postaci długiej (tryb gadatliwy). W naszym systemie grupa wolumenów *rootvg* posiada następującą konfigurację:

```

1  [root@messy ~]# vgdisplay -v rootvg
2      Using volume group(s) on command line
3      Finding volume group "rootvg"
4      --- Volume group ---
5      VG Name                rootvg
6      System ID
7      Format                 lvm2
8      Metadata Areas        1
9      Metadata Sequence No   1
10     VG Access              read/write
11     VG Status              resizable
12     MAX LV                 0
13     Cur LV                 0
14     Open LV                0
15     Max PV                 0
16     Cur PV                 1
17     Act PV                 1
18     VG Size                512.00 MB
19     PE Size                8.00 MB
20     Total PE               64
21     Alloc PE / Size        0 / 0
22     Free PE / Size         64 / 512.00 MB
23     VG UUID                CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v
24
25     --- Physical volumes ---
26     PV Name                /dev/sda7
27     PV UUID                mJDbYE-PipQ-CPDT-miWe-uBjh-bCSb-RYGk0T
28     PV Status              allocatable
29     Total PE / Free PE     64 / 64

```

Istotne informacje zawarte zostały w:

- linii nr 5 – nazwa grupy wolumenów.
- linii nr 13 – liczba utworzonych wolumenów logicznych w obrębie grupy wolumenów.
- linii nr 16 – grupa wolumenów została zbudowana z jednego wolumenu fizycznego.
- linii nr 18 – rozmiar grupy wolumenów (dostępnych w niej partycji fizycznych (ekstendów)) wynosi 512 MB.
- linii nr 19 – rozmiar partycji fizycznej (fizycznego ekstendu) wynosi 8 MB.
- linii nr 20 – grupa wolumenów posiada 64 partycje fizyczne (fizyczne ekstendy).
- linii nr 21 – liczba partycji fizycznych (fizycznych ekstendów) wykorzystanych do utworzenia wolumenu logicznego wynosi 0, zatem ich łączny rozmiar również wynosi 0.



linii nr 22 – w grupie wolumenów pozostało niewykorzystanych 64 partycje fizyczne (ekstendy fizyczne), a ponieważ rozmiar każdego z nich wynosi 8 MB, więc pozostało wolnych 512 MB.

linii nr 23 – identyfikator grupy wolumenów. Jest on wykorzystywany praktycznie podczas operacji przenoszenia grup wolumenów między systemami komputerowymi.

od linii 25 – opis wolumenów fizycznych wchodzących w skład grupy składający się z: ścieżki dostępu do pliku reprezentującego urządzenie fizyczne, identyfikatora wolumenu, jego statusu oraz liczby partycji fizycznych (fizycznych ekstendów) udostępnianych przez wolumen do grupy wolumenów i liczbę partycji nie wykorzystanych.

Mając utworzoną grupę wolumenów, a w niej nie wykorzystane partycje fizyczne, tworzymy wolumen logiczny. Nazwy komend służących do zarządzania wolumenami logicznymi zaczynają się od liter *lv* od Logical Volume, zaś utworzenie to *create*. Stąd wykorzystamy komendę *lvcreate*. Jej wywołanie wymaga podania co najmniej rozmiaru tworzonego wolumenu (wartość opcji **-L**) jego nazwy (wartość opcji **-n**). Jako argument podaje się nazwę grupy wolumenów w obrębie której wolumen logiczny ma zostać utworzony:

```
1 [root@messy ~]# lvcreate -n lv01 -L300M rootvg
2   Rounding up size to full physical extent 304.00 MB
3   Logical volume "lv01" created
```

Wykonanie komendy *lvcreate* spowodowało utworzenie wolumenu logicznego o nazwie *lv01*. W katalogu */dev* utworzony został katalog o nazwie takiej jak nazwa grupy wolumenów, a w nim plik o nazwie identycznej z nazwą założonego wolumenu logicznego:

```
1 [root@messy ~]# ls -l /dev/rootvg/lv01
2 lrwxrwxrwx 1 root root 23 2009-08-17 17:04 /dev/rootvg/lv01 -> /dev/mapper/rootvg-lv01
```

Listę dostępnych w systemie wolumenów logicznych udostępnia komenda *lvdisplay*. Spośród licznych opcji najczęściej wykorzystuje się opcję **-v**, która umożliwia uzyskanie informacji w postaci długiej:

```
1 [root@messy ~]# lvdisplay -v
2   Finding all logical volumes
3   --- Logical volume ---
4   LV Name                /dev/rootvg/lv01
5   VG Name                rootvg
6   LV UUID                J11dBx-JnAn-m0RX-qQu9-43ou-N72C-9JKiAI
7   LV Write Access        read/write
8   LV Status              available
9   # open                 0
10  LV Size                 304.00 MB
11  Current LE              38
12  Segments                1
13  Allocation              inherit
14  Read ahead sectors      auto
15  - currently set to     256
16  Block device            253:0
```

Jak widać w systemie utworzony został jeden wolumen logiczny. Jego nazwa to `/dev/rootvg/lv01` (linia 3). Został on utworzony w oparciu o partycje fizyczne dostępne na wolumenach fizycznych znajdujących się w grupie wolumenów *rootvg* (linia 4). Jego rozmiar wynosi 304 MB (linia 9). Do jego utworzenia zużyto 38 partycji fizycznych (ekstendów). Zwróćmy uwagę, że żądany rozmiar wynosił 300 MB. Jednak przy rozmiarze partycji fizycznej wynoszącym 8 MB, możliwe rozmiary wolumenu logicznego stanowią całkowitą wielokrotność tej wartości, gdyż partycja fizyczna nie może zostać podzielona pomiędzy wolumeny logiczne – jest jednostką niepodzielną. System LVM zawsze utworzy wolumen logiczny o rozmiarze będącym wielokrotnością rozmiaru partycji fizycznej jednak nie mniejszym od żadanego. Do jego utworzenia zostało zużytych 38 partycji fizycznych (linia 10).

Kolejnym krokiem jest założenie na utworzonym wolumenie logicznym systemu plików oraz udostępnienie go użytkownikom, co przedstawiono poniżej:

```

1 [root@messy ~]# mke2fs /dev/rootvg/lv01
2 mke2fs 1.40.8 (13-Mar-2008)
3 Filesystem label=
4 OS type: Linux
5 Block size=1024 (log=0)
6 Fragment size=1024 (log=0)
7 77824 inodes, 311296 blocks
8 15564 blocks (5.00%) reserved for the super user
9 First data block=1
10 Maximum filesystem blocks=67633152
11 38 block groups
12 8192 blocks per group, 8192 fragments per group
13 2048 inodes per group
14 Superblock backups stored on blocks:
15     8193, 24577, 40961, 57345, 73729, 204801, 221185
16
17 Writing inode tables: done
18 Writing superblocks and filesystem accounting information: done
19
20 This filesystem will be automatically checked every 29 mounts or
21 180 days, whichever comes first. Use tune2fs -c or -i to override.
22 [root@messy ~]# mkdir /scratch4
23 [root@messy ~]# mount -t auto /dev/rootvg/lv01 /scratch4
24 [root@messy ~]# df /scratch4
25 Filesystem            1K-blocks      Used Available Use% Mounted on
26 /dev/mapper/rootvg-lv01
27                        301467        2062    283841   1% /scratch4

```

Na utworzonym wolumenie logicznym założono system plików *ext2* (linie 1–21). Następnie utworzono katalog */scratch4* (linia 22) i zamontowano w nim nowoutworzony system plików (linia 23). Zwróćmy uwagę, że w wywołaniach odpowiednich funkcji posługujemy się ścieżką dostępu do pliku reprezentującego wolumen logiczny *lv01*. Komenda *df* (linia 24) wypisuje informacje o systemach plików zamontowanych w systemie. Ostatnia linia opisuje nasz testowy system plików, w którym zostało wykorzystanych 1% bloków danych. Jednak udostępnienie systemu plików użytkownikom natychmiast spowoduje, że wykorzystają oni dostępne bloki danych, pozostawiając najczęściej wolne i–węzły:

```

1 [root@messy ~]# df /scratch4
2 Filesystem      1K-blocks      Used Available Use% Mounted on
3 /dev/mapper/rootvg-lv01
4                 301467      301467          0 100% /scratch4
5 [root@messy ~]# df -i /scratch4
6 Filesystem      Inodes    IUsed   IFree IUse% Mounted on
7 /dev/mapper/rootvg-lv01
8                 77824      13   77811    1% /scratch4

```

Jak widać wszystkie bloki danych zostały wykorzystane. Musimy zatem powiększyć system plików. System plików został założony na wolumenie logicznym, a ten został zbudowany z partycji fizycznych dostępnych w grupie wolumenów. Stąd pierwszym krokiem jest sprawdzenie, czy w grupie wolumenów są jeszcze wolne partycje fizyczne. W tym celu uruchamiamy komendę *vgdisplay*:

```

1 [root@messy ~]# vgdisplay -v
2   Finding all volume groups
3   Finding volume group "rootvg"
4   --- Volume group ---
5   VG Name                rootvg
6   System ID
7   Format                 lvm2
8   Metadata Areas         1
9   Metadata Sequence No   2
10  VG Access               read/write
11  VG Status                resizable
12  MAX LV                  0
13  Cur LV                  1
14  Open LV                 1
15  Max PV                  0
16  Cur PV                  1
17  Act PV                  1
18  VG Size                 512.00 MB
19  PE Size                 8.00 MB
20  Total PE                64
21  Alloc PE / Size         38 / 304.00 MB
22  Free PE / Size          26 / 208.00 MB
23  VG UUID                 CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v
24
25  --- Logical volume ---
26  LV Name                 /dev/rootvg/lv01
27  VG Name                 rootvg
28  LV UUID                 J11dBx-JnAn-mORX-qQu9-43ou-N72C-9JKiAI
29  LV Write Access         read/write
30  LV Status                available
31  # open                  1
32  LV Size                 304.00 MB
33  Current LE              38

```

```

34 Segments          1
35 Allocation        inherit
36 Read ahead sectors auto
37 - currently set to 256
38 Block device      253:0
39
40 --- Physical volumes ---
41 PV Name            /dev/sda7
42 PV UUID            mJDbYE-PipQ-CPDT-miWe-uBjh-bCSb-RYGk0T
43 PV Status          allocatable
44 Total PE / Free PE 64 / 26

```

Najistotniejsza dla nas informacja została zapisana w linii 22. W grupie wolumenów *rootvg* mamy jeszcze dostępnych 26 partycji fizycznych (fizycznych ekstendów). Rozmiar każdego z nich to 8 MB, co daje 208 MB. Pierwszym krokiem jest zatem zwiększenie rozmiaru partycji logicznej, na której znajduje się nasz system plików. Nazwa komendy, którą w tym celu wykorzystamy zaczyna się literami *lv*, a ponieważ chodzi o zmianę rozmiaru, więc drugi człon nazwy stanowi słowo *resize*. Komenda *lvresize* umożliwia zarówno zwiększenie, jak i zmniejszenie rozmiaru wolumenu logicznego, co również jest sporadycznie wykorzystywane. Komendą alternatywną jest komenda *lvextend*, umożliwiająca jedynie zwiększanie rozmiaru wolumenu logicznego. Chcąc zwiększyć rozmiar wolumenu logicznego, musimy jako wartość opcji **-L** podać o jaką wartość rozmiar ten ma zostać zwiększony (zmieniony) oraz jako argumentu użyć nazwy wolumenu. Ponieważ w grupie wolumenów mamy jeszcze dostępne 208 MB, wolumen logiczny *lv01* powiększymy o ten rozmiar:

```

1 [root@messy ~]# lvextend -L+208M /dev/rootvg/lv01
2 Extending logical volume lv01 to 512.00 MB
3 Logical volume lv01 successfully resized

```

Rozmiar wolumenu logicznego, na którym znajduje się system plików został już powiększony. Kolejnym krokiem jest zatem zwiększenie rozmiaru systemu plików. Tu przychodzą nam z pomocą programy narzędziowe. W przypadku systemów plików *ext2* oraz *ext3* jest to program *resize2fs*. Jeśli jako argument zostanie podana nazwa urządzenia, to dokona ono zwiększenia rozmiaru plików do maksymalnego możliwego, czyli do rozmiaru partycji logicznej. Uwaga techniczna dotyczy odmontowania systemu plików, którego rozmiar ma zostać zmieniony, gdyż starsze wersje jądra nie potrafią wykonać tej operacji na zamontowanym systemie plików. Może również okazać się konieczne wykonanie komendy *e2fsck* z opcją **-f**:

```

1 [root@messy ~]# resize2fs /dev/rootvg/lv01
2 resize2fs 1.40.8 (13-Mar-2008)
3 Filesystem at /dev/rootvg/lv01 is mounted on /scratch4; on-line resizing required
4 old desc_blocks = 2, new_desc_blocks = 2
5 resize2fs: Kernel does not support online resizing
6 [root@messy ~]# umount /scratch4
7 [root@messy ~]# resize2fs /dev/rootvg/lv01
8 resize2fs 1.40.8 (13-Mar-2008)
9 Please run 'e2fsck -f /dev/rootvg/lv01' first.
10
11 [root@messy ~]# e2fsck -f /dev/rootvg/lv01

```

```

12 e2fsck 1.40.8 (13-Mar-2008)
13 Pass 1: Checking inodes, blocks, and sizes
14 Pass 2: Checking directory structure
15 Pass 3: Checking directory connectivity
16 Pass 4: Checking reference counts
17 Pass 5: Checking group summary information
18 /dev/rootvg/lv01: 13/77824 files (7.7% non-contiguous), 311296/311296 blocks
19 [root@messy ~]# resize2fs /dev/rootvg/lv01
20 resize2fs 1.40.8 (13-Mar-2008)
21 Resizing the filesystem on /dev/rootvg/lv01 to 524288 (1k) blocks.
22 The filesystem on /dev/rootvg/lv01 is now 524288 blocks long.

```

Po zamontowaniu systemu plików stwierdzamy, że zarówno liczba dostępnych i-węzłów jak i bloków danych uległy zwiększeniu. Przedstawia to poniższy listing:

```

1 [root@messy ~]# mount /dev/rootvg/lv01 /scratch4
2 [root@messy ~]# df /scratch4
3 Filesystem            1K-blocks      Used Available Use% Mounted on
4 /dev/mapper/rootvg-lv01
5                      507748       301723    185054   62% /scratch4
6 [root@messy ~]# df -i /scratch4
7 Filesystem            Inodes      IUsed    IFree IUse% Mounted on
8 /dev/mapper/rootvg-lv01
9                      131072        13   131059    1% /scratch4
10

```

Po pewnym czasie eksploatacji systemu okaże się, że dostępne w systemie plików bloki danych uległy wyczerpaniu. Pierwszą czynnością jest sprawdzenie, czy w grupie wolumenów, w której został utworzony został wolumen logiczny, na którym założyliśmy system plików znajdują się jeszcze wolne partycje fizyczne. Wynik uruchomienia komendy *vgdisplay* został zamieszczony poniżej:

```

1 [root@messy ~]# vgdisplay
2 --- Volume group ---
3 VG Name                rootvg
4 System ID
5 Format                  lvm2
6 Metadata Areas          1
7 Metadata Sequence No    3
8 VG Access                read/write
9 VG Status                resizable
10 MAX LV                   0
11 Cur LV                   1
12 Open LV                  1
13 Max PV                   0
14 Cur PV                   1
15 Act PV                   1
16 VG Size                  512.00 MB
17 PE Size                  8.00 MB

```

18	Total PE	64
19	Alloc PE / Size	64 / 512.00 MB
20	Free PE / Size	0 / 0
21	VG UUID	CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v

Jak wynika z linii 18, całkowita liczba dostępnych partycji fizycznych w grupie wolumenów wynosi 64. Wykorzystanych zostało 64 (linia 19), stąd dostępnych jest 0 (linia 20). Zatem jedynym rozwiązaniem jest dodanie do grupy wolumenów nowego wolumenu fizycznego, który dostarczy nowych partycji fizycznych. W systemie dostępne jest urządzenie fizyczne `/dev/sdc` zasoby którego nie zostały jeszcze wykorzystane. Programem *fdisk* założymy na nim jedną partycję obejmującą cały jego obszar, a następnie komendą *pvcreeate* przekształcimy ją do wolumenu fizycznego. Kolejnym krokiem będzie rozszerzenie istniejącej grupy wolumenów *rootvg* o utworzony wolumen fizyczny. Ponieważ operacji rozszerzenia będzie poddawana grupa wolumenów, więc nazwa wykorzystanej komendy będzie zaczynać się od liter *vg* a następnie pojawi się *extend*. Komenda *vgextend* wymaga podania co najmniej dwóch argumentów. Pierwszym z nich jest nazwa grupy wolumenów, która ma zostać rozszerzona, a kolejnymi nazwy wolumenów fizycznych wyrażone ścieżkami dostępu do plików je reprezentujących, które mają zostać dołączone do grupy. Operacje te zostały przedstawione poniżej:

```

1 [root@messy ~]# fdisk /dev/sdc
2 ....
3 Command (m for help): w
4 The partition table has been altered!
5
6 Calling ioctl() to re-read partition table.
7 Syncing disks.
8 [root@messy ~]# pvcreate /dev/sdc1
9   Physical volume "/dev/sdc1" successfully created
10 [root@messy ~]# vgextend rootvg /dev/sdc1
11 Volume group "rootvg" successfully extended

```

Sytuacja w grupie wolumenów *rootvg* wygląda teraz następująco:

```

1 [root@messy ~]# vgdisplay -v rootvg
2   Using volume group(s) on command line
3   Finding volume group "rootvg"
4   --- Volume group ---
5   VG Name                rootvg
6   System ID
7   Format                  lvm2
8   Metadata Areas          2
9   Metadata Sequence No    4
10  VG Access                read/write
11  VG Status                resizable
12  MAX LV                   0
13  Cur LV                   1
14  Open LV                  1
15  Max PV                   0
16  Cur PV                   2

```

```

17  Act PV                2
18  VG Size               14.74 GB
19  PE Size               8.00 MB
20  Total PE              1887
21  Alloc PE / Size       64 / 512.00 MB
22  Free PE / Size        1823 / 14.24 GB
23  VG UUID               CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v
24
25  --- Logical volume ---
26  ...
27  --- Physical volumes ---
28  PV Name                /dev/sda7
29  PV UUID                mJDbYE-PipQ-CPDT-miWe-uBjh-bCSb-RYGk0T
30  PV Status               allocatable
31  Total PE / Free PE     64 / 0
32
33  PV Name                /dev/sdc1
34  PV UUID                Ml3qvz-KetG-JnEn-jFct-1Ez1-OnI4-Mzk6GK
35  PV Status               allocatable
36  Total PE / Free PE     1823 / 1823

```

Jak widać w skład grupy wolumenów wchodzi teraz dwa wolumeny fizyczne (linia 16) i oba są aktywne (linia 17). Dostępna w obrębie grupy wolumenów przestrzeń adresowa to 14.74 GB (linia 18). Całkowita liczba partycji fizycznych (fizycznych ekstendów), którymi dysponuje grupa wolumenów wynosi 1887 (linia 20). Liczba partycji fizycznych wykorzystanych do budowy wolumenów logicznych to 64 (linia 21). Tworzą one wolumen logiczny, na którym został utworzony system plików zamontowany w katalogu */scratch4*. Istotna jest wiadomość zawarta w linii 22. W grupie wolumenów mamy dostępne 1823 partycje fizyczne. Każda z nich ma rozmiar 8 MB, a więc do dyspozycji pozostaje 14.24 GB. Informacje dotyczące wolumenów logicznych utworzonych w obrębie grupy wolumenów *rootvg* zostały pominięte, gdyż od momentu utworzenia wolumenu logicznego *lv01* oraz zwiększenia jego rozmiaru, w tym zakresie nic nie uległo zmianie. Trzecia sekcja listingu zawiera informacje o wolumenach fizycznych wchodzących w skład grupy wolumenów. Jak widać, w chwili obecnej grupę wolumenów tworzą dwa wolumeny fizyczne */dev/sda7* oraz */dev/sdc1*. Wolumen */dev/sda7* posiada 64 partycje fizyczne, które zostały w całości wykorzystane, zaś dodany do grupy wolumenów wolumen */dev/sdc1* posiada 1823 partycje fizyczne, które są dostępne.

Wiemy już, że mając w obrębie grupy wolumenów niewykorzystane partycje fizyczne możemy wykorzystać je do zwiększenia rozmiaru istniejących partycji logicznych. W naszym przypadku wolne partycje fizyczne pochodzą z dodanego do grupy wolumenów wolumenu */dev/sdc1*. Chcąc rozszerzyć istniejący wolumen logiczny *lv01* musimy wykorzystać partycje fizyczne pochodzące z nowo dodanego wolumenu fizycznego. Prosta operacja rozszerzenia wolumenu logicznego doprowadzi do sytuacji, w której tworzące go partycje fizyczne będą pochodzić z różnych wolumenów fizycznych. Została ona przedstawiona w części teoretycznej wraz z praktycznym omówieniem komend umożliwiających jej przeprowadzenie. Postawmy zatem bardziej interesujące zadanie, polegające na przeniesieniu wolumenu logicznego *lv01* na wolumen fizyczny */dev/sdc1*. Po przeprowadzeniu tej operacji, żadna partycja fizyczna pochodząca z wolumenu fizycznego */dev/sda7* nie będzie wykorzystywana do budowy wolumenu logicznego. Będzie zatem spełniony warunek konieczny i wystarczający do usunięcia wolumenu fizycznego z grupy wolumenów.

Proponowana operacja przesunięcia zawartości ma dotyczyć zawartości wolumenu fizycznego.

Stąd nazwa komendy będzie zaczynać się od *pv*. Ponieważ jest to przesunięcie zawartości wolumenu fizycznego, więc drugi człon nazwy to *move*. Komenda *pvmove* wymaga podania dwóch argumentów. Pierwszym z nich jest nazwa wolumenu fizycznego, z którego zawartość ma zostać przeniesiona, zaś drugą nazwa docelowego wolumenu fizycznego. W naszym przypadku proces przenoszenia miał następujący przebieg:

```

1 [root@messy ~]# pvmove /dev/sda7 /dev/sdc1
2   /dev/sda7: Moved: 29.7%
3   /dev/sda7: Moved: 59.4%
4   /dev/sda7: Moved: 89.1%
5   /dev/sda7: Moved: 100.0%

```

Po jego zakończeniu charakterystyka grupy wolumenów *rootvg* jest następująca:

```

1 [root@messy ~]# vgdisplay -v rootvg
2   Using volume group(s) on command line
3   Finding volume group "rootvg"
4   --- Volume group ---
5   VG Name                rootvg
6   System ID
7   Format                 lvm2
8   Metadata Areas        2
9   Metadata Sequence No  7
10  VG Access              read/write
11  VG Status              resizable
12  MAX LV                 0
13  Cur LV                 1
14  Open LV               1
15  Max PV                 0
16  Cur PV                 2
17  Act PV                 2
18  VG Size                14.74 GB
19  PE Size                8.00 MB
20  Total PE               1887
21  Alloc PE / Size        64 / 512.00 MB
22  Free PE / Size         1823 / 14.24 GB
23  VG UUID                CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v
24
25  --- Logical volume ---
26  LV Name                /dev/rootvg/lv01
27  VG Name                rootvg
28  LV UUID                J11dBx-JnAn-mORX-qQu9-43ou-N72C-9JKiAI
29  ...
30  --- Physical volumes ---
31  PV Name                /dev/sda7
32  PV UUID                mJDbYE-PipQ-CPDT-miWe-uBjh-bCSb-RYGk0T
33  PV Status              allocatable
34  Total PE / Free PE     64 / 64
35

```



```

36 PV Name           /dev/sdc1
37 PV UUID           Ml3qvz-KetG-JnEn-jFct-1Ez1-OnI4-Mzk6GK
38 PV Status         allocatable
39 Total PE / Free PE 1823 / 1759

```

Oczywiście zasoby grupy wolumenów nie uległy zmianie (linie 8–14). Jednak wykorzystanie partycji fizycznych uległo zmianie. Całkowita liczba partycji fizycznych pochodzących z wolumenu `/dev/sda7` wynosi 64, ale w chwili obecnej żadna z tych partycji nie jest wykorzystana. Liczba partycji fizycznych dostępnych na wolumenie fizycznym `/dev/sdc1` wynosi 1823. W chwili obecnej pozostało wolnych 1759 partycji. Zawartość zajętych 64 partycji pochodzi z wolumenu fizycznego `/dev/sda7`. Zwróćmy uwagę, iż operacja przeniesienia zawartości partycji fizycznych (fizycznych ekstendów) została przeprowadzona wówczas, gdy istniejący na nich system plików był zamontowany, a więc dostępny dla użytkowników.

Z grupy wolumenów `rootvg` usuniemy wolumen fizyczny `/dev/sda7`. Operacja odbywa się na poziomie grupy wolumenów. Stąd nazwa komendy będzie rozpoczynać się literami *vg*. Ponieważ dotyczy ona zredukowania zawartości grupy wolumenów, więc drugą część nazwy stanowi słowo *reduce*. Komenda `vgreduce` wymaga podania co najmniej dwóch argumentów. Pierwszym jest nazwa grupy wolumenów, z której mają zostać usunięte wolumeny fizyczne, zaś kolejnymi nazwy usuwanych wolumenów. W przypadku grupy wolumenów `rootvg` operacja usunięcia wolumenu fizycznego `/dev/sda7` ma następujący przebieg:

```

1 [root@messy ~]# vgreduce rootvg /dev/sda7
2 Removed "/dev/sda7" from volume group "rootvg"

```

Po jej wykonaniu charakterystyka grupy wolumenów jest następująca:

```

1 [root@messy ~]# vgdisplay -v rootvg
2   Using volume group(s) on command line
3   Finding volume group "rootvg"
4   --- Volume group ---
5   VG Name           rootvg
6   System ID
7   Format            lvm2
8   Metadata Areas    1
9   Metadata Sequence No 8
10  VG Access          read/write
11  VG Status           resizable
12  MAX LV             0
13  Cur LV             1
14  Open LV            1
15  Max PV             0
16  Cur PV             1
17  Act PV             1
18  VG Size            14.24 GB
19  PE Size            8.00 MB
20  Total PE           1823
21  Alloc PE / Size    64 / 512.00 MB
22  Free PE / Size     1759 / 13.74 GB
23  VG UUID            CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v

```

```

24      --- Logical volume ---
25      ....
26      --- Physical volumes ---
27      PV Name               /dev/sdc1
28      PV UUID               M13qvz-KetG-JnEn-jFct-1Ez1-OnI4-Mzk6GK
29      PV Status              allocatable
30      Total PE / Free PE    1823 / 1759
31

```

Jak widać, teraz grupa wolumenów jest zbudowana w oparciu o jeden wolumen fizyczny */dev/sdc1*. Dostępne w jego obrębie 1759 partycje fizyczne (ekstendy fizyczne) umożliwią zwiększenie jego rozmiaru o 13.74 GB.

Po lekturze rozdziału, operacja zwiększania rozmiaru wolumenu fizycznego i znajdującego się na nim systemu plików nie powinna rodzić problemów. Może jednak zdarzyć się, że zaproponowany rozmiar partycji fizycznej i utworzonego na niej systemu plików jest zbyt duży. Jak zatem „odzyskać” niewykorzystywaną przestrzeń dyskową? Teoretycznie należałoby po pierwsze zmniejszyć rozmiar systemu plików, a następnie rozmiar partycji logicznej, na której ów system plików się znajduje. Problem polega na tym, że programy narzędziowe służące zmianie rozmiaru systemu plików wymagają podania nowego rozmiaru systemu w kilo, mega lub giga bajtach, zaś jednostką rozmiaru partycji logicznej jest rozmiar partycji fizycznej wykorzystanej do jej budowy. Stąd zmianę rozmiaru systemu plików zaleca się przeprowadzać w trzech krokach:

1. Wykorzystując odpowiedni program narzędziowy zmniejszamy rozmiar systemu plików do rozmiaru nie większego niż całkowita wielokrotność rozmiaru partycji fizycznej, z której został zbudowany wolumen logiczny, na którym ten system plików założono.
2. Następnie zmniejszamy rozmiar partycji logicznej tak, aby pozostał on większy od rozmiaru znajdującego się na niej systemu plików z dokładnością do rozmiaru partycji fizycznej.
3. W celu optymalnego wykorzystania przestrzeni dyskowej, zwiększamy rozmiar systemu plików do rozmiaru partycji logicznej, na której jest on założony.

W naszym przypadku system plików zamontowany w katalogu */scratch4* wykorzystuje 64 partycje fizyczne o rozmiarze 8 MB, czyli 512 MB udostępniając 507748 bloków danych o rozmiarze 1 kB, czyli 495.84 MB przestrzeni adresowej dla danych. Założmy, że chcemy ograniczyć rozmiar systemu plików do 452 MB. Postępując zgodnie z przedstawionym algorytmem, zmniejszamy rozmiar systemu plików do żądanej wartości:

```

1  [root@messy ~]# resize2fs /dev/rootvg/lv01 452M
2  resize2fs 1.40.8 (13-Mar-2008)
3  Filesystem at /dev/rootvg/lv01 is mounted on /scratch4; on-line resizing required
4  On-line shrinking from 524288 to 462848 not supported.
5  [root@messy ~]# umount /dev/rootvg/lv01
6  [root@messy ~]# resize2fs /dev/rootvg/lv01 452M
7  resize2fs 1.40.8 (13-Mar-2008)
8  Please run 'e2fsck -f /dev/rootvg/lv01' first.
9
10 [root@messy ~]# e2fsck -f /dev/rootvg/lv01
11 e2fsck 1.40.8 (13-Mar-2008)
12 Pass 1: Checking inodes, blocks, and sizes

```

```

13 Pass 2: Checking directory structure
14 Pass 3: Checking directory connectivity
15 Pass 4: Checking reference counts
16 Pass 5: Checking group summary information
17 /dev/rootvg/lv01: 13/131072 files (7.7% non-contiguous), 318263/524288 blocks
18 [root@messy ~]# resize2fs /dev/rootvg/lv01 452M
19 resize2fs 1.40.8 (13-Mar-2008)
20 Resizing the filesystem on /dev/rootvg/lv01 to 462848 (1k) blocks.
21 The filesystem on /dev/rootvg/lv01 is now 462848 blocks long.

```

W przypadku starszych wersji jądra może okazać się, że konieczne jest odmontowanie systemu plików, którego rozmiar ma zostać zmniejszony (linia 4, 5). Następnie konieczne jest uruchomienie komendy *e2fsck* z opcją *-f*. Dokona ona m.in. przeniesienia zajmowanych bloków danych na początek listy. Zmniejszając rozmiar systemu plików zwolnimy partycje fizyczne z jego końca, na których znajdują się bloki danych. Stąd operacja ta teoretycznie zapobiega utracie danych (linie 10–17). Następnie zmniejszamy rozmiar systemu plików do żadanego. Po wykonaniu tej operacji możemy zwolnić partycje fizyczne, które nie są wykorzystywane przez struktury systemu plików, a wchodzi w skład wolumenu logicznego, na którym jest on utworzony. Pytanie ile partycji fizycznych zwolnić, lub jaki powinien być nowy rozmiar partycji logicznej? Zwróćmy uwagę, że w chwili obecnej lista bloków danych systemu plików posiada 462848 bloki o rozmiarze 1 kB. Zajmuje ona 56.5 partycji fizycznych, gdyż partycja fizyczna posiada rozmiar 8192 kB. Stąd proponowany rozmiar wolumenu fizycznego będzie wynosił 58 partycji fizycznych, czyli 464 MB. Mamy zatem:

```

1 [root@messy ~]# lvresize -L 464M /dev/rootvg/lv01
2 WARNING: Reducing active logical volume to 464.00 MB
3 THIS MAY DESTROY YOUR DATA (filesystem etc.)
4 Do you really want to reduce lv01? [y/n]: y
5 Reducing logical volume lv01 to 464.00 MB
6 Logical volume lv01 successfully resized

```

Po wykonaniu operacji zmniejszenia rozmiaru jedyne wolumenu fizycznego utworzonego w obrębie grupy wolumenów *rootvg* jej charakterystyka wygląda następująco:

```

1 [root@messy ~]# vgdisplay rootvg
2 --- Volume group ---
3 VG Name                rootvg
4 System ID
5 Format                  lvm2
6 Metadata Areas          1
7 Metadata Sequence No    9
8 VG Access               read/write
9 VG Status               resizable
10 MAX LV                  0
11 Cur LV                  1
12 Open LV                 0
13 Max PV                  0
14 Cur PV                  1
15 Act PV                  1

```

```

16 VG Size                14.24 GB
17 PE Size                8.00 MB
18 Total PE              1823
19 Alloc PE / Size       58 / 464.00 MB
20 Free PE / Size        1765 / 13.79 GB
21 VG UUID                CPsy19-zZ1R-k2qd-vwb1-Tv1l-HqVD-wSMh0v

```

Zgodnie z oczekiwaniami liczba wykorzystanych partycji fizycznych (ekstendów fizycznych) wynosi 58, co daje 464 MB. Ostatnim krokiem jest zwiększenie rozmiaru plików do dostępnego na wolumenie logicznym *lv01* miejsca:

```

1 [root@messy ~]# resize2fs /dev/rootvg/lv01
2 resize2fs 1.40.8 (13-Mar-2008)
3 Resizing the filesystem on /dev/rootvg/lv01 to 475136 (1k) blocks.
4 The filesystem on /dev/rootvg/lv01 is now 475136 blocks long.

```

Rozmiar zmniejszonego systemu plików jest w chwili obecnej następujący:

```

1 [root@messy ~]# df /dev/rootvg/lv01
2 [root@messy ~]# df /scratch4
3 Filesystem            1K-blocks      Used Available Use% Mounted on
4 /dev/mapper/rootvg-lv01
5                        460144      301723    148919   67% /scratch4
6 [root@messy ~]# df -i /scratch4
7 Filesystem            Inodes      IUsed      IFree IUse% Mounted on
8 /dev/mapper/rootvg-lv01
9                        118784        13    118771     1% /scratch4

```

Uwaga praktyczna: przed zmniejszeniem rozmiaru systemu plików należy wykonać komendę *df* w celu sprawdzenia, ile bloków danych jest w systemie plików niewykorzystanych. Rozmiar systemu plików możemy zmniejszyć nie więcej niż o rozmiar partycji fizycznych, na których znajdują się niewykorzystane bloki danych.

LVM dostarcza także możliwości łatwego przenoszenia grup wolumenów pomiędzy systemami. Należy pamiętać, aby przed usunięciem grupy z systemu odmontować wszystkie systemy plików założone na jej partycjach oraz uczynić ją nieaktywną. Służy do tego komenda *vgchange*, którą w tym wypadku należy użyć z opcją **-a** i przełącznikiem *n*. Jako argument podajemy nazwę grupy wolumenów. Następnie informujemy system, że grupa wolumenów zostanie z niego usunięta. W tym celu wykorzystujemy komendę *vgexport* podając jako argument nazwę usuwanej grupy wolumenów. Operacja udostępnienia zewnętrznej grupy wolumenów w innym systemie wymaga wykonania komendy *vgimport* z argumentem będącym nazwą grupy wolumenów, a następnie korzystając z komendy *vgchange* z opcją **-a** i przełącznikiem *y*.

### 3.7 Programowe macierze dyskowe

Pomysł reprezentowania w systemie komputerowym wielu dysków fizycznych jako jeden zasób logiczny ma na celu zwiększenie wydajności lub/i podniesienie bezpieczeństwa przechowywanych danych. Zasada działania polega na rozdzieleniu operacji wejścia/wyjścia pomiędzy dyski

z wykorzystaniem techniki przeplotu. Realizacja praktyczna może polegać na zbudowaniu odpowiedniego kontrolera jako urządzenia fizycznego, który sterować będzie pracą dysków. Innym rozwiązaniem jest wykorzystanie zwykłych kontrolerów zrealizowane przez odpowiedni sterownik. W pierwszym przypadku mamy do czynienia z macierzami *sprzętowymi*. W drugim z *programowymi* i to one będą przedmiotem naszego zainteresowania.

### 3.7.1 Podstawy teoretyczne

Pierwsze informacje o sposobach łączenia dysków w macierze RAID (ang. *Redundant Array of Inexpensive Disks*) można znaleźć w [6]. Poniżej krótko omówiono poziomy zaproponowane w przytoczonej artykule, jak również te, które zostały zaproponowane ostatnio, a obecnie są szeroko wykorzystywane.

#### RAID 0 – paskowanie (stripping)

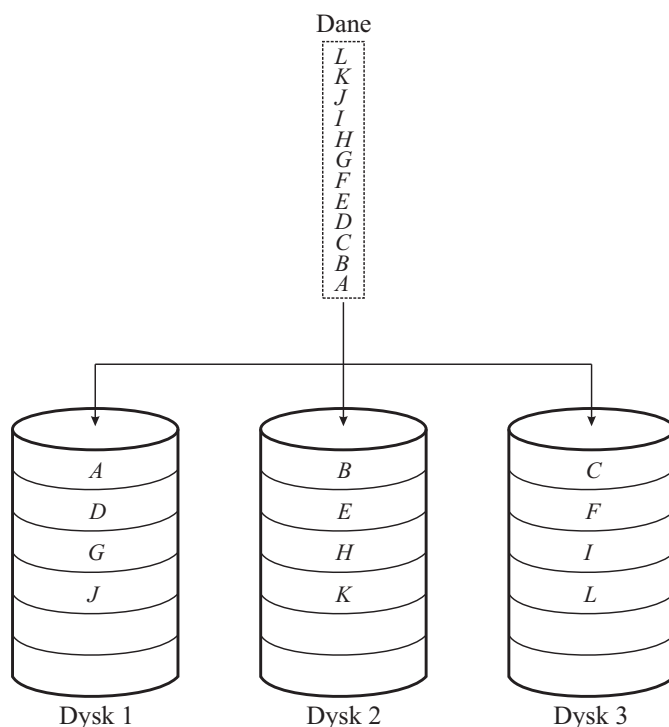
RAID 0 stanowi najpopularniejszy i jednocześnie najprostszy w implementacji poziom macierzy dyskowych. Jednak ze względu na brak jakichkolwiek zabezpieczeń danych, przez niektórych nie jest on uznawany za prawdziwą konfigurację macierzy RAID. Poziom RAID 0 może obejmować dowolną ilość dysków, które rozpatrywane są łącznie jako pojedyncza struktura dyskowa. Widoczny w ten sposób jeden dysk, zostaje podzielony na paski (ang. *stripe*), w taki sposób, aby dane były rozpraszane równomiernie pomiędzy wszystkie dyski. Taka struktura umożliwia równomierny rozkład obciążenia każdego z dysków. Podczas operacji zapisu, dane zostają podzielone na fragmenty (bloki) a następnie pierwszy blok zapisywany jest na pierwszym dysku, drugi na drugim i tak dalej, aż zostaną zapisane wszystkie dane. Operacja zapisu zostaje więc przeprowadzona równolegle co powoduje, że zapis dokonywany jest znacznie szybciej niż zapis na pojedynczym dysku. Dyski do tworzenia macierzy RAID 0 powinny być jednakowej pojemności i najlepiej, jeśli są też jednakowej wydajności. W przypadku różnic w pojemnościach, większe dyski zostają pomniejszone do pojemności najmniejszego z nich poprzez wyłączenie z użytku bloków powyżej danego zakresu. Niektóre kontrolery RAID potrafią wykorzystać tę wolną przestrzeń w razie ewentualnej awarii. Taka awaria jednak musi dotyczyć nie całego dysku, a jedynie jego fragmentów. Wówczas dane zostają zapisane na wolnej przestrzeni. Metoda ta zwiększa minimalnie bezpieczeństwo, jednak w praktyce częściowe uszkodzenie dysku oznacza, że bardzo szybko należy się spodziewać uszkodzenia jego kolejnych bloków.

Sposób zapisu danych na dyskach macierzy RAID poziomu 0 przedstawiono na rysunku 3.43.

W konfiguracji macierzy RAID 0 duży wpływ na wydajność ma wybór rozmiaru „pasków” na jakie zostanie podzielony wolumin logiczny, składający się dysków znajdujących się w macierzy. W przypadku gdy rozmiar pliku będzie mniejszy niż rozmiar wybranego paska, przyrost wydajności może być nawet zerowy. Wynika to z faktu, iż cały plik zmieści się w jednym fragmencie na dysku logicznym co oznacza, że fizycznie zostanie zapisany tylko na jednym dysku. W najlepszym wypadku, gdy dane zostaną idealnie podzielone na fragmenty tak, żeby przy jednokrotnym podziale można było zapisać cały plik na wszystkich dyskach (po jednym fragmencie na każdy dysk), uzyskany wzrost wydajności może być nawet  $n$ -krotny, gdzie „ $n$ ” oznacza ilość dysków w macierzy.

Obecnie każdy dostępny kontroler RAID posiada możliwość założenia na podłączonych dyskach RAID poziomu 0, ze względu na najprostszą ze wszystkich poziomów implementację. Założona na kilku dyskach macierz tego poziomu nie może zostać zmniejszona, gdyż każdy dysk w macierzy stanowi niezbędną (i nie zdublowaną) część z całej struktury. Wiele kontrolerów pozwala jednak na dodawanie nowych dysków do macierzy.

Zależność wydajności w macierzy RAID 0 nie wzrasta liniowo wraz ze zwiększeniem liczby dysków. Wzrost wydajności jest mniejszy także przy większej ilości dysków, ze względu na narzut



Rysunek 3.43: Organizacja danych na dyskach macierzy RAID poziomu 0.

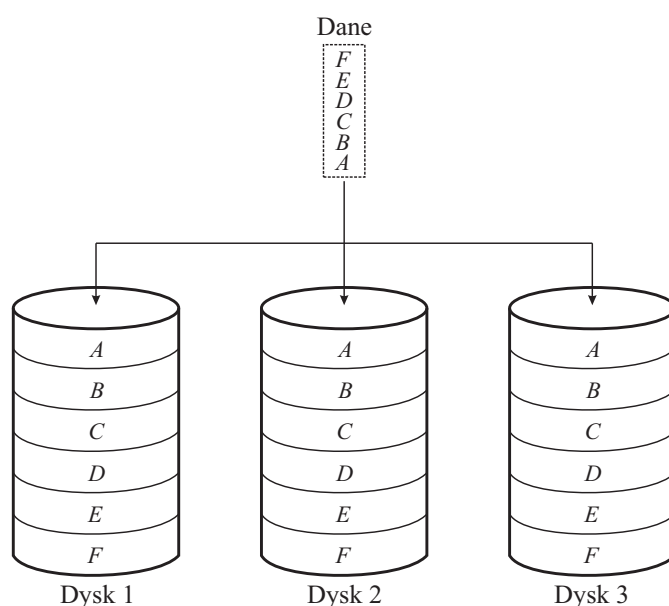
związany z operacjami przetwarzania danych do zapisu oraz danych już odczytanych. Procentowy wzrost wydajności dla 20 dysków nie będzie tak duży jak procentowy wzrost dla 10 dysków. Macierz RAID 0 poprawia uniwersalnie wydajność podsystemu plików. Nieliniowy wzrost wydajności związany jest także z tym, że dyski w macierzy nie obracają się synchronicznie, to znaczy, że każdy dysk obraca się niezależnie od pozostałych. Wiele zależy także od kontrolera RAID lub programowego rozwiązania, które nie zawsze są w stanie obsłużyć sporą ilość operacji zapisu lub odczytu. Ważny jest także sam sposób połączenia dysków nie tylko między sobą ale i z kontrolerem. Przy obecnych prędkościach dysków, po połączenie kilkunastu dysków o dużych prędkościach i wydajnościach, wąskim gardłem może okazać się sama magistrala danych.

### RAID 1 – kopia lustrzana (mirroring)

Poziom RAID 1 zwany inaczej dublowaniem, został opracowany aby wyeliminować wady poziomu RAID 0. Jest to jednocześnie najprostsza wersja macierzy z nadmiarowością danych przez co jest często stosowana. Zasadą RAID 1 jest zapisywanie danych w dwóch miejscach praktycznie jednocześnie, dzięki czemu uzyskujemy lustrzaną kopie naszych danych. Dublowanie polega na tym, że każdemu dyskowi w macierzy odpowiada/przyporządkowany jest co najmniej jeden dysk, nazywany dyskiem lub kopią lustrzaną. Z tego dysku, w razie awarii można odtworzyć dane. Jeśli awarii ulegnie dysk składowy, system przechodzi w tryb pracy jak dla zwykłego dysku, w tym czasie należy wymienić uszkodzony dysk na nowy, a dzięki istnieniu dokładnej kopii uszkodzonego dysku, zawartość odpowiadającego mu dysku lustrzanego zostaje po prostu skopiowana na nowy dysk. Operacja kopiowania nazywana resynchronizacją wykonywana jest zwykle z mniejszymi

prędkościami (a przynajmniej nie z maksymalnymi) w celu zmniejszenia obciążenia systemu. Czasem nie wymagamy pełnego kopiowania danych na nowy dysk, np. w przypadku gdy dysk pracował w macierzy chwilowo. Wykonywana wtedy resynchronizacja nazywa się resynchronizacją różnicową i polega na kopiowaniu/przywróceniu tylko zmienionych danych. Ze względu na to, iż wszystkie zapisywane dane posiadają automatycznie tworzone kopie bezpieczeństwa, RAID 1 odporny jest na awarie  $N-1$  dysków, gdzie „ $N$ ” oznacza liczbę dysków znajdujących się w macierzy. Prawdopodobieństwo awarii macierzy zmniejsza się więc proporcjonalnie z liczbą dysków macierzy. Przykładowo, jeśli w macierzy do której użyto trzech trzech dysków, dwa z nich zostaną uszkodzone, to nie stracimy żadnych danych, a system będzie normalnie działał. To dzięki pozostałemu jednemu dyskowi, który zawiera całe dane.

Sposób zapisu danych na dyskach macierzy RAID poziomu 1 przedstawiono na rysunku 3.44.



Rysunek 3.44: Organizacja danych na dyskach macierzy RAID wykorzystującej mechanizm kopii lustrzanej.

Z punktu widzenia wydajności RAID 1 nie jest ani szybki ani wolny. Ta konfiguracja praktycznie nie zmienia wydajności macierzy podczas odczytu i podczas zapisu danych. Obecne implementacje sprzętowe na kontrolerach RAID bez problemów radzą sobie z prostym mechanizmem tworzenia kopii lustrzanych danych. Wykonywany podczas zapisu danych odczyt weryfikujący dane z drugiego dysku często odbywa się „w tle” zupełnie nie obciążając systemu.

Na ogół w RAID 1 stosuje się dublowanie dwudrożne. Można jednak zastosować dublowanie trójdrożne. Dublowanie więcej niż trójdrożne jest bardzo rzadko stosowane, gdyż liczba dysków potrzebnych do takiej konfiguracji może zostać wykorzystana do konfiguracji innych poziomów RAID.

W niewielkich, prostych systemach z niewielką liczbą dysków RAID 1 stanowi dobre i stosunkowo tanie zabezpieczenie. Jednak tak jak i w RAID 0, tak i tutaj niebezpieczna może okazać się awaria samego kontrolera. Istnieje możliwość tzw. dupleksowania, tzn. podłączania dwóch kontrolerów do dwóch macierzy. Jednak takie konfiguracje nie są już atrakcyjne cenowo i dlatego, gdy potrzebne są lepsze zabezpieczenia stosuje się inne poziomy RAID.

## RAID 2 – przeplot bitowy z redundancją

RAID 2 jest poziomem oferującym ochronę danych przed błędami. Jednak ze względu na bardzo skomplikowaną implementację (złożoność algorytmu do generowania kodów Hamminga) oraz na ograniczenia dotyczące ilości dysków oraz ich organizacji w macierzy, RAID 2 jest używany bardzo rzadko. RAID 2 przypomina RAID 0 ze względu na stosowany przeplot danych między dyskami. Jednak zamiast przeplotu blokowego występującego w RAID 0, w RAID 2 występuje przeplot bitowy. Poziom ten wymaga kilku dysków do przechowania danych nadmiarowych, co zmniejsza powierzchnię dyskową możliwą do efektywnego wykorzystania. Dla 14 dysków w macierzy aż 4 z nich wymagane są na dane nadmiarowe, przy 32 dyskach z danymi, na dane nadmiarowe potrzebne jest aż 7 dysków.

W celu pełnego opisu macierzy RAID 2, należy przedstawić sposób obliczania kodów Hamminga.

Metoda obliczania kodów Hamminga jest jedną ze stosowanych metod do wykrywania błędów za pomocą bitów parzystości. Pierwszym etapem jest określenie liczby jedynek w binarnej reprezentacji danych. Gdy liczba ta jest parzysta, bit parzystości przyjmuje wartość 1, a gdy nieparzysta, wartość bitu parzystości ustalana jest na 0. Dzięki takiemu zliczaniu liczby bitów, jeden bit parzystości pozwala na wykrycie przekłamania pojedynczego bitu, jednak bez informacji o tym który bit jest nieprawdziwy. Tak ustalony 1-bitowy kod parzystości nazywa się „kodem o odległości 2”. Z tej zależności widać, że w momencie, gdy zmieniają się dwa dowolne bity, lub jeden bit danych wraz z bitem parzystości, to taka konfiguracja nadal nie wykaze błędu.

Dlatego w celu wykrywania większej ilości błędów niż jeden, należy stosować „kod o odległości 3”. Dzięki temu staje się możliwe wykrycie i skorygowanie błędu jednego bitu, oraz wykrycie ale bez możliwości skorygowania błędu dwóch bitów. Analogicznie, dla 64 bitów danych, wymagane jest 7 bitów parzystości, dla 128 bitów potrzebne jest 8 bitów parzystości.

W 1950 roku Richard Hamming opracował i opublikował algorytm znany dzisiaj jako Kod Hamminga. Odległością Hamminga dwóch ciągów bitowych o takiej samej długości definiuje się jako liczbę pozycji (miejsc), na których te dwa ciągi się różnią. Oznaczamy ją jako  $d(x1, x2)$ . Gdy w algorytmie zastosujemy większą liczbę bitów służących do kontroli poprawności i gdy różne błędy powodują powstanie różnej kombinacji bitów, to możliwe staje się wskazanie w którym miejscu nastąpiła zmiana. Gdy zbiór danych jest siedmiobitowy, to teoretycznie trzy dodatkowe bity są wystarczające do wskazania pozycji na której wystąpił błąd. W swoich algorytmach Hamming chciał zwiększyć maksymalnie odległość przy jednoczesnym zwiększeniu współczynnika ilości informacji zawartych w kodzie.

Ogólny algorytm użycia parzystości dla ogólnego kodu Hamminga przedstawia się następująco:

- pozycje bitów będące potęgami dwójki są użyte jako bity parzystości,
- wszystkie pozostałe pozycje nie będące potęgami 2 to bity danych,
- pozycja bitu parzystości wskazuje, które bity mają być sprawdzane, a które należy opuszczać,
- numeracja pozycji rozpoczyna się od 1, a pierwszym sprawdzanym bitem jest bit na pozycji  $n+1$ ,
  - pozycja 1: opuść: 0 bit, sprawdź: 1 bit,
  - pozycja 2: opuść: 1 bit, sprawdź: 2 bity,
  - pozycja 4: opuść: 3 bity, sprawdź: 4 bity,



- pozycja 8: opuść: 7 bitów, sprawdź: 8 bitów,
- itd.

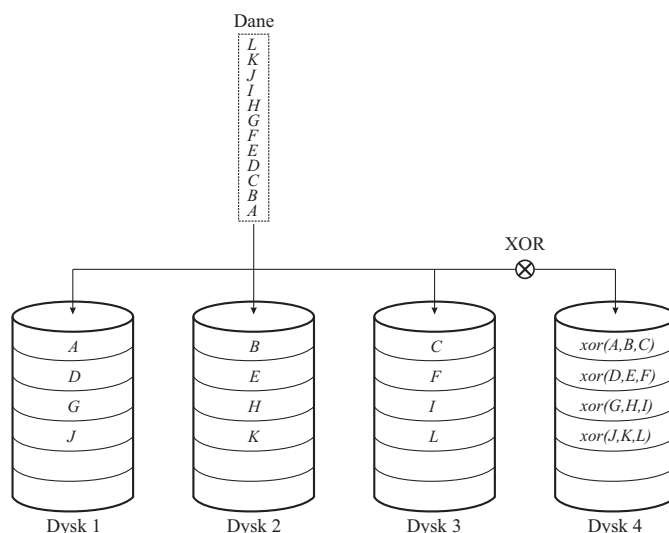
Po zakończeniu transmisji sprawdzany jest tzw. syndrom błędu, któremu przypisuje się wartość 0 gdy błąd nie wystąpił lub różny od 0 gdy znaleziony został błąd.

Jak widać, powyższy algorytm nie jest prosty w implementacji. Jego ograniczenia i duże wymagania, w porównaniu z innymi poziomami RAID sprawiają, że macierze RAID 2 są praktycznie nie stosowane. W praktyce niewielu kontrolerów posiada możliwość zakładania macierzy tego poziomu (przykładowo dostępna w systemach z rodziny Linux komenda *mdadm* nie posiada funkcji zakładania macierzy tego poziomu).

### RAID 3 – pełny przeplot (RAID 0) z dyskiem nadmiarowym

Poziom RAID 3 powstał w celu poprawienia wad, które stwarzały poprzednie poziomy, tzn. niskie bezpieczeństwo danych (RAID 0) oraz wysokie koszty bezpieczeństwa (RAID 1). RAID 3 opiera się na rozwarstwieniu, podobnie jak RAID 0, ale dodatkowo posiada dysk na którym znajdują się wyliczane bity parzystości, inaczej nazywane sumami kontrolnymi. Osobny dysk na dane nadmiarowe zmniejsza w niewielkim stopniu narzuty na przetwarzanie danych podczas zapisu.

Sposób zapisu danych na dyskach macierzy RAID poziomu 3 przedstawiono na rysunku 3.45.



Rysunek 3.45: Organizacja danych na dyskach macierzy RAID poziomu 3.

RAID 3 zachowuje zalety macierzy RAID 0. Charakteryzuje go podobna wydajność w przypadku operacji na danych o dużym rozmiarze i niska wydajność przy operacjach, które dotyczą danych mieszczących się na jednym dysku i niewielkiej części obszaru z dysku drugiego. Implementacje wykorzystywane w RAID 3 do obliczania bitów parzystości są dość wydajne. Szacuje się, że obciążenie generowane przez obliczanie danych nadmiarowych nie przekracza 5%. Podczas odczytu danych, RAID 3 zachowuje się tak jak RAID 0, gdyż dane czytane są tylko z dysków z danymi, a dysk z bitami parzystości nie jest wówczas wykorzystywany.

RAID 3 został zorientowany na optymalizację żądań, które wykorzystują wszystkie dyski znajdujące się w macierzy. Z tego względu, stosuje się tzw. „synchronizację obrotów”, która powoduje wymuszenie dla wszystkich dysków tej samej pozycji w danej chwili czasu. Zmniejsza to

opóźnienie podczas operacji sekwencyjnych. Również jak dla RAID 0 operacje, które wymagają licznych żądań losowych są w macierzach tego poziomu wykonywane mało wydajnie. Podczas żądań losowych poziom RAID 3 może być porównany do jednego szybkiego i bezpiecznego dysku obsługującego żądania dokładnie tak, jak gdyby w macierzy znajdował się jeden dysk.

#### **RAID 4 – Jak poziom 3, ale z przeplotem blokowym**

RAID 4 jest bardzo podobny do opisanego poprzednio poziomu - RAID 3, jednak zamiast podziału na poziomie bitów, w RAID 4 występuje podział na bloki. Tak samo jak w przypadku RAID 3, występują dane nadmiarowe znajdujące się na jednym dysku. Poprzez zmianę sposobu podziału na podział blokowy, RAID 4 nie używa tak małych fragmentów danych jak RAID 3, ale fragmenty wielkościami zbliżone do tych stosowanych w RAID 0. Tutaj także można wybierać rozmiar paska (stripe size), który tak jak i w RAID 0, wpływa na wydajność w zależności od konkretnych zastosowań.

Podobnie jak w RAID 0, poziom 4 oferuje słabą wydajność zapisu i odczytu dla operacji na danych o niewielkim rozmiarze, tzn dla małych plików. Wydajność jest wtedy zbliżona, a nawet niższa niż w przypadku pojedynczego dysku. Dla dużych plików oferuje jednak bardzo dobrą wydajność odczytu sekwencyjnego. Jest on porównywalny z osiąganym w RAID 0, przy niewielkich wymaganiach na dane redundantne, które zawsze zajmują tylko jeden dysk. Z powodu braku synchronizacji obrotów dysków, operacje zapisu są w RAID 4 wolniejsze niż w RAID 3. Aktualizacje danych na dysku nadmiarowym wymaga odszukania na nim samym odpowiednich bloków danych, co powoduje, że dysk z kodami parzystości tak jak i w RAID 3, jest wąskim gardłem macierzy. Z tego też powodu, RAID 4 znajduje zastosowanie przede wszystkim w systemach, w których operacje odczytu są częstsze niż operacje zapisu.

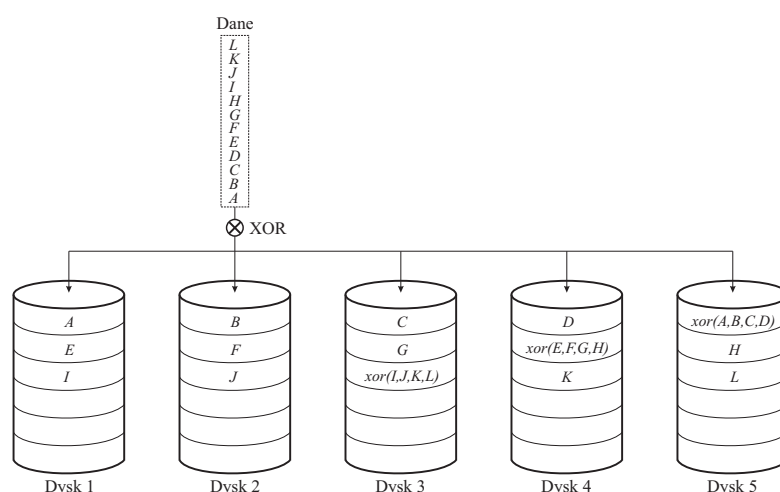
#### **RAID 5 – Jak poziom 4 ale z rozproszoną sumą kontrolną**

RAID 5 jest rozszerzeniem koncepcji powstałej w RAID 3, której główną wadą było nie uwzględnianie specyfiki przetwarzania wielozadaniowego. W RAID 5 problem ten został rozwiązany poprzez umieszczenie danych nadmiarowych na wszystkich dyskach z podziałem tak, jak w RAID 4 - na poziomie bloków. Ze względu na występujące w macierzy operacje tworzenia sum kontrolnych, do utworzenia macierzy RAID 5 wymagane są co najmniej 3 dyski.

Organizację danych na dyskach macierzy RAID poziomu 5 zbudowanej na pięciu dyskach przedstawiono na rysunku 3.46.

Jak widać, gdy blok parzystości paska  $n$  w macierzy składającej się z 5 dysków znajduje się na dysku 5, to blok parzystości paska  $n + 1$  zostanie umieszczony na dysku numer 4. I tak dalej, aż do momentu, gdy pasek zostanie umieszczony na dysku numer 1, co spowoduje, że cykl przeplotu bloków parzystości rozpocznie się od nowa.

Wydajność macierzy RAID 5 często jest niewłaściwie rozumiana. Podczas odczytu jednowątkowego z macierzy składającej się z  $N$  dysków, odczyt jest prawie tak samo szybki jak odczyt z macierzy RAID 0 składającej się z  $N - 1$  dysków. Odjęcie jednego dysku spowodowane jest tym, że równowartość pojemnościowa jednego dysku w RAID 5 przeznaczona jest na dane nadmiarowe. Podobnie jak w przypadku RAID 0, tak i tutaj, maksymalną wydajność osiąga się wtedy, gdy nasze żądanie jest ułożone na granicy pasków, a wielkość żądania jest wielokrotnością grubości paska. W wyniku tego każdy pasek odczytywany jest z innego dysku (przynajmniej jej początkowe bloki), co jest najbardziej pożądaną sytuacją. Również jak w RAID 0, najmniejsza wydajność występuje wtedy, gdy żądanie jest stosunkowo niewielkie, a obejmuje fizycznie 2 dyski. Powoduje to, że uaktywniane są aż trzy dyski, w tym jeden z danymi nadmiarowymi. Optymalna konfiguracja macierzy RAID 5, to taka, w której znajduje się od 4 do 15 dysków. Większa niż



Rysunek 3.46: Organizacja danych na dyskach macierzy RAID poziomu 5 zbudowanej na pięciu dyskach.

15 liczba dysków, powoduje zmniejszenie wydajności całej macierzy. Jednak najważniejszym elementem jest redukcja ilości danych nadmiarowych, które w przypadku 15 dysków, stanowią już tylko  $\frac{1}{15}$  ilości przechowywanych danych.

Zapis w macierzach RAID 5 jest bardziej skomplikowany dlatego, że każda operacja, wywołuje tak naprawdę cykl odczyt-modyfikacja-zapis. Odczyt obejmuje całą warstwę danych (odpowiadających zapisywanemu blokowi innych bloków danych), w celu obliczenia sumy kontrolnej.

W praktyce operacje zapisu mogą być bardziej złożone i trudniejsze do prostego opisanie. Konfiguracja kontrolera lub oprogramowania które będzie zarządzać macierzą nie jest łatwe i podobnie jak w konfiguracji RAID 0, tak i tutaj wiele czynników wpływa na dobór odpowiedniej konfiguracji. Podstawowym celem jest dostosowanie konfiguracji do przeznaczenia, a więc wielkości przechowywanych plików, co przekłada się na wielkość żądań które będziemy wykonywać. Wielkość warstwy danych najlepiej ustawić na wielkość typowej operacji I/O. Przykładowo, gdy typowa operacja będzie miała rozmiar 128 KB, a macierz składa się z 5 dysków, fragment danych powinien mieć rozmiar 26 KB:

$$\text{wielkość członu} = \frac{\text{wielkość typowej operacji}}{\text{liczba dysków} - 1} \text{ I/O}$$

Macierze RAID 5 umożliwiają rekonstrukcje w razie awarii jednego z dysków. Rekonstrukcja wymusza odczyt wszystkich dysków i na tej podstawie wyznaczenia sumy kontrolnej. Odbudowywanie macierzy powoduje, że każdy odczyt dysku wymaga aż  $2 \cdot \frac{(n-1)}{n}$  odczytów, podczas gdy w trybie normalnej pracy jest to tylko  $n$  odczytów. Większość, szczególnie nowych kontrolerów, umożliwia zapis do macierzy podczas rekonstrukcji danych. Jednak taki zapis jest o wiele wolniejszy niż w czasie normalnego trybu pracy macierzy.

Obecnie w konfiguracjach RAID 5, stosuje się tzw. inteligentne kontrolery, które potrafią wykryć prawdopodobne uszkodzenia dysku i wymusić tym samym rekonstrukcję macierzy. Niektóre kontrolery potrafią analizować stopień obciążenia dysków, oraz wykrywać najbardziej podejrzane obszary dysku i odpowiednio przenosić dane tak, aby eliminować zbieranie się większych operacji dyskowych na tych samych obszarach fizycznych.

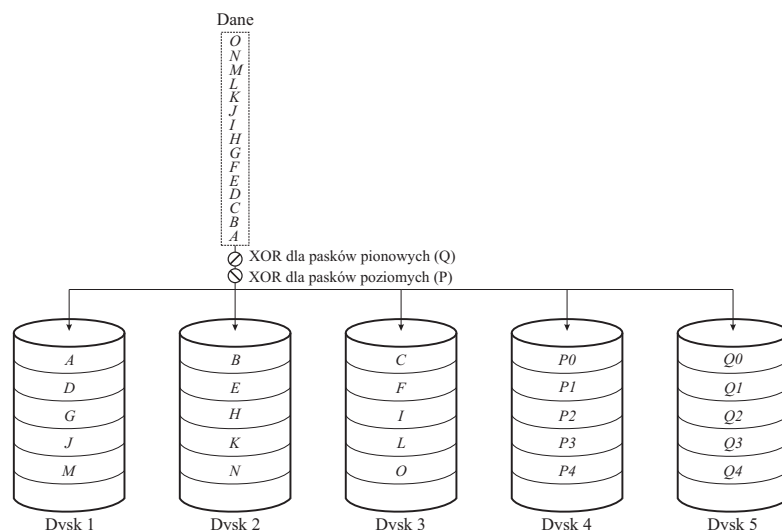
Poziom 5 macierzy RAID jest bardzo popularny i często stosowany jako recepta na idealną kombinację wydajności, pojemności i bezpieczeństwa. Dlatego znajduje on zastosowanie w sys-

temach ogólnego przeznaczenia, a dzięki możliwości konfiguracji macierzy i wpływania na jej wydajność poprzez dobór kilku parametrów, także w systemach z konkretnymi specyfikacjami i odpowiednio ustalonymi wymaganiami jak serwery bazodanowe i transakcyjne.

### RAID 6 – dyski niezależne z dublowaniem parzystości

Kolejny poziom RAID 6 najbardziej przypomina poziom RAID 5 z zaimplementowaną dodatkową metodą rozpraszania nadmiarowych, nazywaną schematem Reeda–Solomona. Podobnie jak w RAID 5, dane nadmiarowe przechowywane są na wszystkich dyskach. Występująca różnica jest taka, że ta konfiguracja wymaga do tego dwóch, a nie jednego dodatkowego dysku. Dzięki takiej strukturze, macierz zachowuje wysoką wydajność (jak w RAID 5), przy jednoczesnym zwiększeniu bezpieczeństwa przechowywanych danych, gdyż pozwala na obsłużenie awarii więcej niż jednego dysku. W RAID 6 wyliczane są dwie sumy kontrolne. W zależności od konkretnej implementacji, pierwsza obliczana jest tak samo jak w RAID 5, czyli za pomocą operacji logicznej XOR. Druga suma kontrolna może wykorzystywać algorytm Reeda–Solomona. Niektóre firmy wykorzystują też własne rozwiązania i sposoby obliczania danych parzystości drugiego typu. Wydajnościowo RAID 6 jest zbliżony do RAID 5 jeśli chodzi o odczyt danych. Jest natomiast wolniejszy podczas zapisu.

Graficznie organizacja danych na dyskach macierzy RAID 6 zaproponowanej przez firmę IBM Almaden, nazwanej *EvenOdd* została przedstawiona na rysunku 3.47.



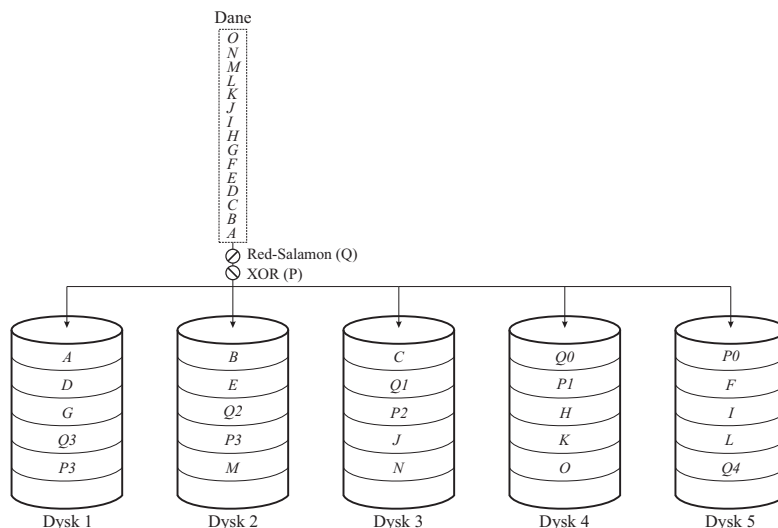
Rysunek 3.47: Schemat organizacji danych na dyskach macierzy RAID 6 o nazwie *EvenOdd* składającej się z pięciu dysków.

Implementacja *EvenOdd* polega na użyciu podwójnego kodowania z wykorzystaniem operacji XOR. Pierwsze kodowanie wykonywane jest na blokach poziomych, a drugie na blokach diagonalnych. Taka implementacja jednak posiada bardzo niską wydajność zapisu.

W implementacji wykorzystującej algorytm Reeda–Solomona do pierwszego kodowania wykorzystywana jest funkcja XOR. Podobnie jak w RAID 5, dane nadmiarowe rozmieszczone są na wszystkich dyskach. Do drugiego kodowania używa się algorytmu Reeda–Solomona, a obliczone w ten sposób dane zapisywane są również na wszystkich dyskach. Główną wadą tej metody są wysokie wymagania odnośnie kontrolera. Liczba dysków zawierających drugie dane nadmiarowe (te

obliczone algorytmem Reeda–Solomona) może być dowolnie rozszerzona, a tym samym zwiększa się ilość dysków które mogą ulec awarii.

Graficzna organizacja danych na dyskach macierzy RAID 6 wykorzystująca algorytm Reeda–Solomona została przedstawiona na rysunku 3.48.

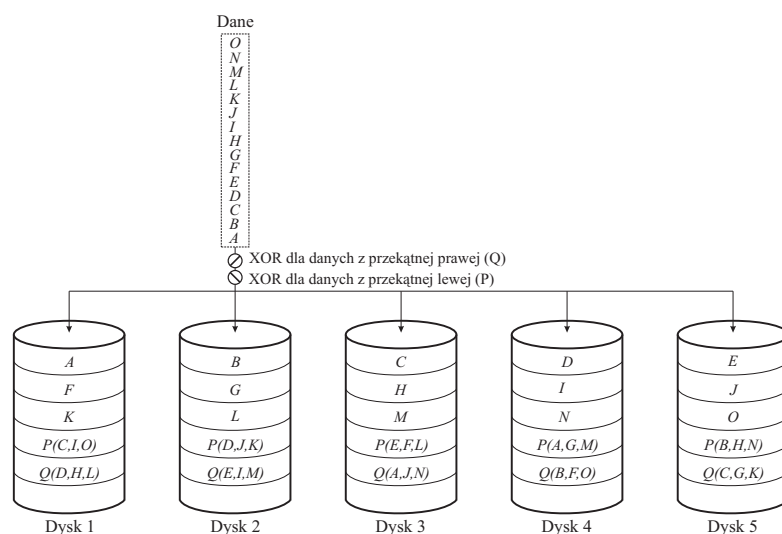


Rysunek 3.48: Organizacja danych na dyskach macierzy RAID 6 wykorzystująca algorytm Reeda–Solomona składająca się z pięciu dysków.

Kolejna implementacja poziomu RAID 6 nazywa się X–Code. W tej wersji sumy kontrolne obliczane są po przekątnych. Pierwsza wartość XOR obliczana jest od lewej do prawej, druga, od prawej do lewej. Niestety ten schemat obliczeń może być wykorzystywany tylko wtedy, gdy liczba dysków jest liczbą pierwszą. Graficznie organizacja danych na dyskach macierzy RAID 6 nazywanej X–Code została przedstawiona na rysunku 3.49.

Szacuje się, że najczęściej konfiguracje dyskowe opierają się na RAID 5, gdzie dane są chronione na wypadek uszkodzenia jednego z dysków. W razie awarii, proces odbudowy trwać może bardzo długo (od kilku godzin do kilka/kilkanaście dni w zależności od liczby dysków, ich pojemności oraz wydajności kontrolera lub systemu). Jednak podczas rekonstrukcji macierzy, dane w niej zapisane nie podlegają żadnej dodatkowej ochronie, co oznacza, że gdy podczas rekonstrukcji awarii ulegnie kolejny dysk, dane zostają utracone. Przed taką sytuacją potrafi zabezpieczyć poziom RAID 6. Podczas awarii nie ma konieczności natychmiastowej rekonstrukcji macierzy, bo dane i tak są nadal zabezpieczone (w sposób zbliżony jak w RAID 5).

Obecnie konfiguracja RAID 6 nie jest jeszcze bardzo popularna ze względu na spore wymagania wydajnościowe co do kontrolera, a tym samym jego cenę. Jedynie najdroższe kontrolery potrafią dobrze (wydajnie obliczać sumy kontrolne) obsługiwać poziom RAID 6. Mimo to, konfiguracja taka zapewnia bardzo wysokie bezpieczeństwo. Dodatkowo, pomimo drogich kontrolerów, okazuje się tańsza, jeśli chodzi o ilość danych nadmiarowych w stosunku do RAID 1. Tym samym koszt dysków w macierzy jest mniejszy. Aby sensownie wykorzystać wydajność jaką oferuje RAID 6 i zagwarantować danym wysokie bezpieczeństwo, macierz powinna składać się z co najmniej 4 do 5 dysków (minimalna liczba dysków wynosi 4). Obecnie RAID 6 znajduje zastosowanie np. w systemach bankowych, gdzie wymagane jest maksymalne bezpieczeństwo danych. Wiele firm produkuje swoje własne odmiany RAID 6, które zwykle wprowadzają niewielkie zmiany w stosunku do „podstawowego” założenia RAID 6. Często są to rozwiązania własne, opatentowane



Rysunek 3.49: Organizacja danych na dyskach macierzy RAID 6 X-Code.

i dokładny sposób działania (głównie sposób obliczania danych parzystości drugiego typu) nie jest podawany. Przykładem takich implementacji może być stworzona przez firmę NetApp konfiguracja nazwana RAID Dual Parity, w której można ustalić dowolną liczbę dysków parzystości.

Ze względu na coraz szybsze procesory stosowane w komputerach i kontrolerach, RAID 6 może stosować coraz efektywniejsze algorytmy zabezpieczeń. Producenci sprzętu wymyślają coraz to nowsze rozwiązania, których głównym założeniem jest „podwójna parzystość”, a co za tym idzie możliwość zabezpieczenia w przypadku awarii dwóch, a czasem i więcej dysków w macierzy.

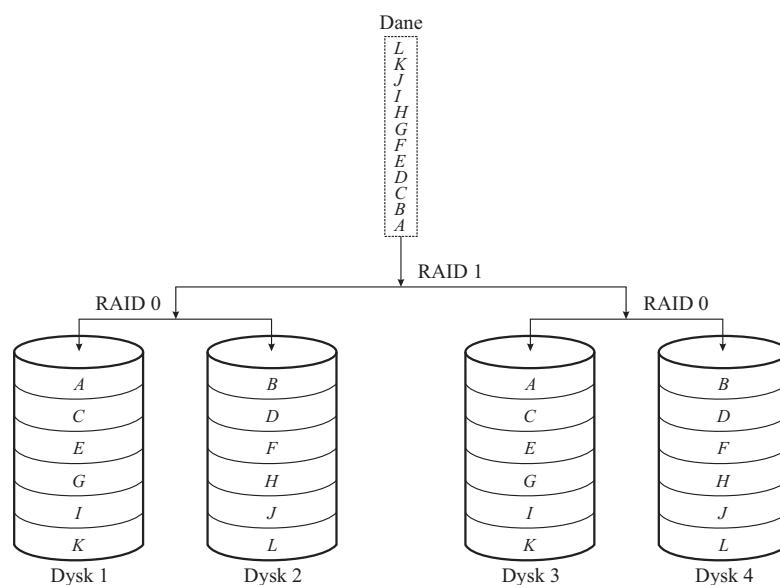
### RAID 0+1 – paskowanie z kopią lustrzaną

Poziom RAID 0+1, zapisywany często jako RAID 01 lub RAID 0/1, to połączenie konfiguracji RAID 0 z konfiguracją RAID 1. W efekcie połączenia uzyskujemy macierz RAID 1, której elementami są macierze RAID 0. Tak utworzona macierz 0+1 posiada wydajność operacji zapisu i odczytu jak macierz RAID 0, oraz zapewnia bezpieczeństwo danych jak RAID 1. Uszkodzenie jednego z dysków powoduje, że macierz RAID 0+1 staje się na czas rekonstrukcji macierzą RAID 0. Główną zaletą takiej konfiguracji jest prostota implementacji. Jednak koszt przechowywania danych jest wysoki, ze względu na sporą ilość dysków wykorzystywanych na dane nadmiarowe. Minimalna ilość dysków potrzebna do utworzenia RAID 0+1 to 4 dyski.

Schematycznie organizacja danych na dyskach macierzy RAID 0+1 została przedstawiona na rysunku 3.50.

### RAID 1+0 – kopia lustrzana z paskowaniem

Poziom RAID 1+0, zapisywany często jako RAID 10, to połączenie zalet i niestety wad konfiguracji RAID 0 oraz RAID 1. Gwarantujące wydajność jak w RAID 0, oraz bezpieczeństwo jak w RAID 1. Jest to konfiguracja odwrotna do opisanej powyżej (RAID 0+1). Tworzenie macierzy RAID 1+0 składa się z dwóch etapów. Pierwszy polega na utworzeniu z dysków konfiguracji RAID 1, a następnie dla wynikowej macierzy stosuje się RAID 0, czyli elementami (dyskami) macierzy RAID 0, są macierze RAID 1. Powstała konfiguracja RAID 1+0 składa się z połączenia



Rysunek 3.50: Organizacja danych na dyskach macierzy 0+1.

dysków lustrzanych, co powoduje, że podczas awarii jednego z dysków, podczas rekonstrukcji macierzy odbudowywany jest tylko fragment całej macierzy.

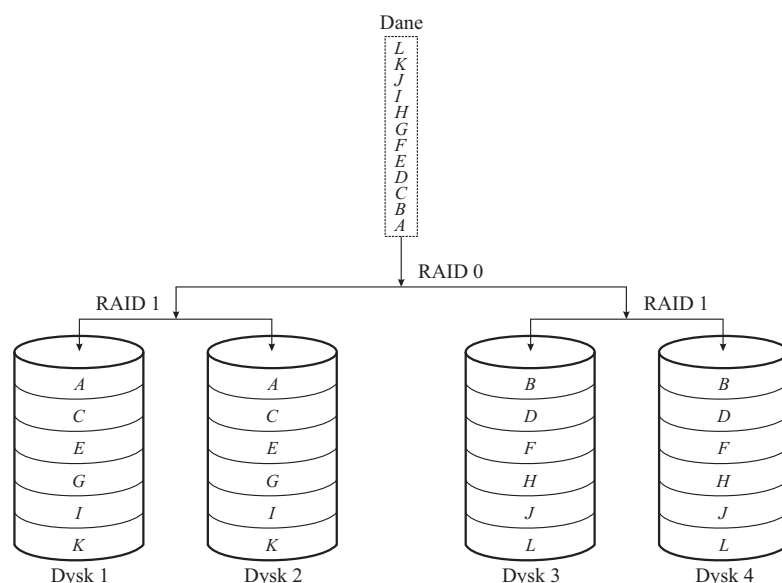
RAID 1+0 zapewnia szybkość macierzy RAID 0, zwiększone bezpieczeństwo RAID 1 (w przypadku wielu dysków, jest możliwość awarii większej liczby dysków niż jeden), oraz stosuje stosunkowo prostą implementację. Dane są po prostu kopiowane na dyski lustrzane, oraz łączone ze sobą, bez stosowania żadnych sum kontrolnych jak np. w poziomach RAID 3 i 5. Jednak w stosunku do macierzy z kodami parzystości, koszt macierzy jest wyższy, ze względu na większą ilość danych (dysków) nadmiarowych.

Przykładowa organizacja danych na dyskach macierzy RAID 1+0 została przedstawiona na rysunku 3.51.

Ze względu na wyższą wydajność, bardziej popularna jest macierz RAID 0+1. Znajduje zastosowanie przede wszystkim w komputerach o dużych wymaganiach co do wydajności operacji dyskowych z zachowaniem wysokiego bezpieczeństwa danych. Druga konfiguracja mieszana (1+0) jest niewiele wolniejsza, ale zapewnia przy tym większe bezpieczeństwo. W celu zrozumienia różnic pomiędzy RAID 0+1 a RAID 1+0 i tego, jak obie konfiguracje wpływają na bezpieczeństwo danych, można posłużyć się schematem z rysunku raid1001.

W konfiguracji 1+0 można zerwać mirroring, zastąpić dysk kopii lustrzanej nowym i odbudować jego zawartość z drugiego dysku. Taka konfiguracja, przetrwa awarię którejś z kopii lustrzanych, oraz awarię poszczególnych dysków w obu macierzach. Oczywiście o ile nie będą to odpowiadające sobie dyski. Dlatego, jak widać na powyższym schemacie, w macierzy składającej się z czterech dysków, gdzie dyski A1B1 są odzwierciedlone na dyskach A2B2, RAID 1+0 może przetrwać równocześnie awarie par dysków np. A1 i B2, lub B1 i A2, o ile nie będą to pary dysków odpowiadających sobie (np. A1 i A2).

W RAID 0+1 dyski najpierw konfigurowane są w RAID 0, a dopiero potem wynik jest poddawany mirroringowi. W razie awarii jednego z dysków z macierzy RAID 0+1, pierwszy utracony dysk oznacza utratę dysku z jednej macierzy RAID 0. Aby rozpocząć rekonstrukcję macierzy, trzeba ją praktycznie zbudować od nowa. Możemy to osiągnąć poprzez odbudowanie przepłotu



Rysunek 3.51: Schematyczna organizacja danych na dyskach macierzy RAID 1+0.

danych z działających dysków w uszkodzonej kopii lustrzanej, albo poprzez zastosowanie wszystkich nowych dysków odpowiadających uszkodzonemu zestawowi.

Stąd wyższość konfiguracji RAID 1+0, która może przetrwać awarię więcej niż jednego dysku. Obie konfiguracje są bardzo zbliżone jeśli chodzi o wydajność i również obie potrzebują tyle samo dodatkowych dysków na dane nadmiarowe.

### RAID – konfiguracje mieszane 5+0, 0+5, 5+1, 1+5

Obecnie większość standardowych kontrolerów oraz oprogramowanie do zarządzania macierzami dyskowymi, posiada możliwość łączenie ze sobą różnych poziomów RAID. Łączenie polega na odpowiednim podziale dysków na grupy. Następnie w każdej grupie zostaje założony odpowiedni poziom RAID, a następnie dla całej grupy stosujemy inny poziom RAID.

Jedną z konfiguracji mieszanych (poza opisanymi już RAID 1+0 oraz RAID 0+1) jest RAID 1+5. W takiej implementacji, najpierw tworzony jest mirroring zestawu dysków, a następnie na takim zestawie wynikowym RAID 5. Taka konfiguracja umożliwia zrównoleglenie odczytu na dwa dyski (dzięki dyskom lustrzanym) oraz możliwość równoległego przeprowadzania zapisu, tak jak ma to miejsce w RAID 1. RAID 1+5 jest konfiguracją o wysokiej wydajności i gwarantuje wysokie bezpieczeństwo danych. Jednak jej koszt, ze względu na tworzone kopie lustrzane, jest wysoki, co powoduje, że często zamiast RAID 1+5 wybierany jest RAID 1+0.

Innym typem konfiguracji mieszanej jest RAID 5+0. Taka konfiguracja umożliwia odbudowę w przypadku awarii pojedynczego dysku z każdej z macierzy składowych (macierze składowe to macierze RAID 5). Konfiguracja ta posiada lepszą wydajność zapisu niż RAID 5. Jednak w razie awarii jednego z dysków, odbudowa jest długotrwała i w dużym stopniu zmniejsza wydajność systemu. RAID 5 używany jest także w konfiguracji mieszanej RAID 0+5. Zasada budowy, podobnie jak w przypadku wszystkich konfiguracji mieszanych, pozostaje taka sama: z dysków tworzy się grupy połączone w RAID 0, a następnie z tak powstałych woluminów dyskowych tworzy się macierz RAID 5.



### 3.7.2 Programowe macierze dyskowe w dystrybucji RedHat

Obsługę macierzy dyskowych w systemach linuxowych zapewnia sterownik o nazwie *md* (*Multiple Devices*). Aktualnie systemy te obsługują następujące programowe macierze dyskowe: RAID0, RAID1, RAID4, RAID5, RAID6 oraz RAID10. Dodatkowo obsługiwany jest także poziom o nazwie LINEAR, który pozwala łączyć miejsce z kilku partycje występujących na różnych urządzeniach fizycznych i udostępniać jako ciągłą przestrzeń adresową. Poziom nazwany MULTIPATH nie ma odpowiednika w rzeczywistych poziomach, a stanowi mechanizm umożliwiający dostęp do spójnej przestrzeni adresowej z wykorzystaniem nazw urządzeń ją tworzących. Jest wykorzystywany w celu umożliwienia dostępu do zapisanych w niej danych w przypadku awarii sprzętu lub zapełnienia fizycznego urządzenia. Podobnie FAULTY, nie jest rzeczywistym poziomem, a jedynie tworzy warstwę nad urządzeniem, umożliwiającą symulowanie awarii.

W systemach dystrybucji RedHat, do tworzenia i administrowania programowymi macierzami dyskowymi służy komenda *mdadm*. W jej wywołaniu należy podać opcjonalnie tryb pracy, ścieżkę dostępu do pliku urządzenia reprezentującego macierz, opcje, jeśli są konieczne, oraz nazwy plików urządzeń tworzących macierz. Zaczniemy od kilku praktycznych przykładów użycia. W testowym systemie mamy dostępne trzy partycje, o różnych rozmiarach i występujące na różnych dyskach. Są to:

1. */dev/sda7* o rozmiarze 530113 bloków,
2. */dev/sdb2* o rozmiarze 1574370 bloków,
3. */dev/sdc2* o rozmiarze 6907950 bloków.

Zatem łącznie mamy dostępnych 9012433 bloki. Utwórzmy z nich ciągłą przestrzeń adresową, a więc wykorzystując poziom LINEAR. Postać komendy jest następująca:

```
1 [root@messy /]# mdadm -C /dev/md0 --level=linear -n 3 /dev/sda7 /dev/sdb2 /dev/sdc2
2 mdadm: array /dev/md0 started.
```

Utworzona macierz dyskowa jest gotowa do pracy. Reprezentuje ją plik o nazwie */dev/md0*. Na wstępie wyjaśnienie nazwy urządzenia. Komenda *mdadm* narzuca pewien standard nazewnictwa. Dla macierzy dyskowych niepartycjonowanych, obsługiwanych przez jądra systemu operacyjnego w wersji 2.4, przyjmuje się, że pliki urządzeń powinny być tworzone według następujących schematów: */dev/mdNN* lub */dev/md/NN*, gdzie *NN* oznacza numer urządzenia. Konwencja ta obowiązuje również dla macierzy obsługiwanych przez jądra w wersji 2.6. Dla macierzy partycjonowanych, obsługiwanych przez jądra w wersji 2.6 przyjmuje się, że może je reprezentować plik o nazwie pasującej do jednego z następujących schematów: */dev/md/dNN* lub */dev/md\_dNN*.

Przyjijmy się charakterystyce utworzonej macierzy:

```
1 [root@messy /]# mdadm -D /dev/md0
2 /dev/md0:
3     Version : 00.90.03
4     Creation Time : Wed Jul 28 19:42:27 2010
5     Raid Level : linear
6     Array Size : 9012160 (8.59 GiB 9.23 GB)
7     Raid Devices : 3
8     Total Devices : 3
9     Preferred Minor : 0
10    Persistence : Superblock is persistent
```

```

11      Update Time : Wed Jul 28 19:42:27 2010
12      State : clean
13      Active Devices : 3
14      Working Devices : 3
15      Failed Devices : 0
16      Spare Devices : 0
17
18      Rounding : 64K
19
20      UUID : a3371490:14f96ffa:1894c650:1232fd22
21      Events : 0.1
22
23
24      Number      Major      Minor      RaidDevice State
25      0           8          7          0        active sync   /dev/sda7
26      1           8          18         1        active sync   /dev/sdb2
27      2           8          34         2        active sync   /dev/sdc2

```

Informacja o poziomie macierzy znajduje się w linii 5 powyższego listingu. Rozmiar, przedstawiony w linii 6, różni się od całkowitej o 273 bloki, które zostały zużyte na struktury danych opisujących utworzoną macierz. Macierz tworzą 3 urządzenia. Ich pełny opis, w postaci tabelki, zamyka informację o stanie macierzy. Istotny jest unikalny numer identyfikacyjny urządzenia zawarty w linii 21. Zapisany na każdym urządzeniu utworzonej macierzy jest wykorzystywany do jej odtwarzania w przypadku awarii lub przeniesienia do innego systemu komputerowego.

Na udostępnianej przez macierz przestrzeni adresowej można założyć system plików, a następnie udostępnić go montując w odpowiednim katalogu:

```

1  [root@messy ~]# mkfs -t xfs -f /dev/md0
2  meta-data=/dev/md0          isize=256    agcount=4, agsize=563260 blks
3      =                      sectsz=512    attr=2
4  data      =                  bsize=4096   blocks=2253040, imaxpct=25
5      =                      sunit=0       swidth=0 blks
6  naming    =version 2        bsize=4096   ascii-ci=0
7  log       =internal log     bsize=4096   blocks=2560, version=2
8      =                      sectsz=512    sunit=0 blks, lazy-count=0
9  realtime  =none             extsz=4096   blocks=0, rtextents=0
10 [root@messy ~]# mount -t xfs /dev/md0 /scratch2
11 [root@messy ~]# df /dev/md0
12 Filesystem          1K-blocks      Used Available Use% Mounted on
13 /dev/md0             9001920        4256   8997664   1% /scratch2

```

Usunięcie urządzenia z macierzy nie jest możliwe dla macierzy każdego poziomu. Można ogólnie przyjąć zasadę, że operacja ta może zostać przeprowadzona, jeśli nie spowoduje trwałej utraty danych. Dla macierzy typu LINEAR nie jest to możliwe. W przypadku poziomu pierwszego RAID, usunięcie jednej kopii danych nie spowoduje ich utraty, a jedynie obniży poziom bezpieczeństwa ich przechowywania. W każdym przypadku istniejącą macierz można rozmontować. Po wykonaniu tej operacji każde z tworzących je urządzeń staje się niezależne. Należy jednak zwrócić uwagę, iż pozostają na nim struktury danych opisujące macierz, do której ono należało.

Informacja ta jest wykorzystywana w nowszych wersjach programu *mdadm* do automatycznego odtwarzania macierzy, np. w przypadku przeniesienia dysków do innego systemu.

Rozmontowanie istniejącej macierzy oraz utworzenie nowej, pracującej na poziomie pierwszym RAID, przebiega w naszym przypadku następująco:

```

1 [root@messy ~]# mdadm -S /dev/md0
2 mdadm: stopped /dev/md0
3 [root@messy ~]# mdadm -C /dev/md0 --level=1 -n 3 /dev/sda7 /dev/sdb2 /dev/sdc2
4 mdadm: /dev/sda7 appears to be part of a raid array:
5     level=raid1 devices=3 ctime=Thu Jul 29 12:47:25 2010
6 mdadm: /dev/sdb2 appears to be part of a raid array:
7     level=raid1 devices=3 ctime=Thu Jul 29 12:47:25 2010
8 mdadm: /dev/sdc2 appears to be part of a raid array:
9     level=raid1 devices=3 ctime=Thu Jul 29 12:47:25 2010
10 Continue creating array? y
11 mdadm: array /dev/md0 started.

```

Utworzenie macierzy poziomu pierwszego RAID trwa znacznie dłużej niż pozostałych poziomów, a to ze względu na konieczność synchronizacji zawartości urządzeń ją tworzących. Utworzona macierz ma następującą charakterystykę:

```

1 [root@messy ~]# mdadm -D /dev/md0
2 /dev/md0:
3     Version : 00.90.03
4     Creation Time : Thu Jul 29 13:48:38 2010
5     Raid Level : raid1
6     Array Size : 530048 (517.71 MiB 542.77 MB)
7     Used Dev Size : 530048 (517.71 MiB 542.77 MB)
8     Raid Devices : 3
9     Total Devices : 3
10    Preferred Minor : 0
11    Persistence : Superblock is persistent
12
13    Update Time : Thu Jul 29 13:48:38 2010
14    State : clean
15    Active Devices : 3
16    Working Devices : 3
17    Failed Devices : 0
18    Spare Devices : 0
19
20    UUID : 1a44a54d:1da7f18c:b42dcceb:94d910a0
21    Events : 0.1
22
23    Number   Major   Minor   RaidDevice State
24     0         8       7         0     active sync  /dev/sda7
25     1         8      18         1     active sync  /dev/sdb2
26     2         8      34         2     active sync  /dev/sdc2

```

Zwróćmy uwagę na jej rozmiar. Jest on o 65 bloków dyskowych mniejszy od rozmiaru najmniejszej partycji (urządzenia) wykorzystanego do jej budowy (w naszym przypadku */dev/sda7*).

Jest to rozmiar maksymalnego, możliwego do utworzenia na każdej partycji obszaru pomniejszony o fragment konieczny do przechowania struktur opisujących. Na partycjach o większym rozmiarze, wchodzących w skład macierzy, pozostałe miejsce jest tracone.

W celu usunięcia dowolnego urządzenia należy w pierwszej kolejności zaznaczyć je jako uszkodzone:

```

1 [root@messy ~]# mdadm /dev/md0 -f /dev/sdb2
2 mdadm: set /dev/sdb2 faulty in /dev/md0
3 [root@messy ~]# mdadm -D /dev/md0
4 /dev/md0:
5     Version : 00.90.03
6     Creation Time : Thu Jul 29 13:48:38 2010
7     Raid Level : raid1
8     Array Size : 530048 (517.71 MiB 542.77 MB)
9     Used Dev Size : 530048 (517.71 MiB 542.77 MB)
10    Raid Devices : 3
11    Total Devices : 3
12 Preferred Minor : 0
13    Persistence : Superblock is persistent
14
15    Update Time : Thu Jul 29 15:14:35 2010
16    State : clean, degraded
17    Active Devices : 2
18    Working Devices : 2
19    Failed Devices : 1
20    Spare Devices : 0
21
22    UUID : 1a44a54d:1da7f18c:b42dcceb:94d910a0
23    Events : 0.6
24
25    Number   Major   Minor   RaidDevice State
26      0       8       7         0    active sync   /dev/sda7
27      1       0       0         1    removed
28      2       8      34         2    active sync   /dev/sdc2
29
30      3       8      18         -    faulty spare   /dev/sdb2

```

Jak widać (linie 10 oraz 11), całkowita liczba urządzeń tworzących macierz wynosi 3, natomiast w chwili obecnej aktywne są jedynie dwa (linie 17, 18) i jedno uszkodzone (linia 19). Uszkodzone urządzenie można z macierzy usunąć. Oczywiście rozpoczynamy od zmian w strukturach opisujących macierz:

```

1 [root@messy ~]# mdadm /dev/md0 -r /dev/sdb2
2 mdadm: hot removed /dev/sdb2
3 [root@messy ~]# mdadm -D /dev/md0
4 /dev/md0:
5     Version : 00.90.03
6     Creation Time : Thu Jul 29 13:48:38 2010
7     Raid Level : raid1

```

```

8      Array Size : 530048 (517.71 MiB 542.77 MB)
9      Used Dev Size : 530048 (517.71 MiB 542.77 MB)
10     Raid Devices : 3
11     Total Devices : 2
12     Preferred Minor : 0
13     Persistence : Superblock is persistent
14
15     Update Time : Thu Jul 29 15:22:14 2010
16     State : clean, degraded
17     Active Devices : 2
18     Working Devices : 2
19     Failed Devices : 0
20     Spare Devices : 0
21
22     UUID : 1a44a54d:1da7f18c:b42dcceb:94d910a0
23     Events : 0.10
24
25     Number   Major   Minor   RaidDevice State
26     0         8       7       0         active sync  /dev/sda7
27     1         0       0       1         removed
28     2         8      34       2         active sync  /dev/sdc2

```

Teraz urządzenie można fizycznie usunąć, a następnie w jego miejsce zainstalować nowe oraz dodać je do macierzy. W naszym przypadku będzie to ta sama partycja:

```

1  [root@messy ~]# mdadm /dev/md0 -a /dev/sdb2
2  mdadm: re-added /dev/sdb2
3  [root@messy ~]# mdadm -D /dev/md0
4  /dev/md0:
5      Version : 00.90.03
6      Creation Time : Thu Jul 29 13:48:38 2010
7      Raid Level : raid1
8      Array Size : 530048 (517.71 MiB 542.77 MB)
9      Used Dev Size : 530048 (517.71 MiB 542.77 MB)
10     Raid Devices : 3
11     Total Devices : 3
12     Preferred Minor : 0
13     Persistence : Superblock is persistent
14
15     Update Time : Thu Jul 29 15:30:55 2010
16     State : clean, degraded, recovering
17     Active Devices : 2
18     Working Devices : 3
19     Failed Devices : 0
20     Spare Devices : 1
21
22     Rebuild Status : 11% complete
23
24     UUID : 1a44a54d:1da7f18c:b42dcceb:94d910a0

```

25	Events : 0.14					
26						
27	Number	Major	Minor	RaidDevice	State	
28	0	8	7	0	active sync	/dev/sda7
29	3	8	18	1	spare rebuilding	/dev/sdb2
30	2	8	34	2	active sync	/dev/sdc2

Dodanie urządzenia do istniejącej macierzy skutkuje koniecznością synchronizacji jego zawartości z pozostałymi urządzeniami. Stan macierzy, zamieszczony na powyższym listingu, sporządzono w czasie trwania procesu synchronizacji. Liczba urządzeń macierzy wynosi 3 (linie 10 i 11). Liczba urządzeń aktywnych to 2 (linia 17) mimo, iż działających jest 3 (linia 18). Wyjaśnienie znajduje się w linii 16 opisującej stan macierzy. Stopień zaawansowania operacji opisuje linia 22.

Utworzona programem *mdadm* macierz dyskowa będzie aktywna do momentu zamknięcia systemu operacyjnego. Po jego ponownej inicjalizacji nie zostanie uruchomiona, gdyż informacje o jej konfiguracji nie zostały nigdzie zapisane. Na dyskach lub partycjach wchodzących w jej skład zostały zapisane struktury ją opisujące, które mogą posłużyć do inicjalizacji. Opis konfiguracji macierzy dyskowych działających w danym systemie komputerowym zawarty jest w pliku o nazwie *mdadm.conf* znajdującym się w systemach rodziny RedHat w katalogu */etc*, lub w katalogu */etc/mdadm*. Jest to plik tekstowy, jak mówi podręcznik systemowy, zbudowany ze słów oddzielonych co najmniej jednym białym znakiem. Linie puste są ignorowane. Dopuszcza się występowanie komentarza. Ma on charakter linikowy, od pojawienia się znaku do znaku przejścia do nowej linii. Każda linia rozpoczynająca się znakiem spacji lub tabulacji jest traktowana jako kontynuacja poprzedniej. Pozostałe linie muszą rozpoczynać się słowem kluczowym (lub jego skrótem utworzonym z 3 pierwszych liter). Mogą pojawić się następujące słowa kluczowe:

- **DEVICE** – linia rozpoczynająca się tym słowem kluczowym służy do zdefiniowania urządzeń (dysków lub partycji), z których w systemie zostaną zbudowane macierze. Linia może zawierać nazwy kilku urządzeń, oddzielonych co najmniej jednym białym znakiem. Nazwy urządzeń mogą być specyfikowane z wykorzystaniem symboli uogólnień (zdefiniowanych w rozdziale 7 podręcznika systemowego jako **glob**). Zamiast listy nazw urządzeń, w linii tej może pojawić się słowo **partitions**. W takim przypadku program *mdadm* przeczyta z pliku */proc/partitions* numery *minor* oraz *major* plików urządzeń wszystkich opisanych w nim partycji. Następnie z ich wykorzystaniem oraz na podstawie zawartości katalogu */dev* określi ścieżki dostępu do reprezentujących je plików. Przykłady:

```
DEVICE /dev/sda7 /dev/sdb2 /dev/sdc2
```

```
DEV /dev/sda7 /dev/sd[bc]2
```

```
DEVICE /dev/hda*
```

- **ARRAY** – linia służy do definiowania macierzy. Wymagane jest, aby drugim słowem była nazwa pliku urządzenia reprezentującego macierz. Kolejne słowa określają cechy charakterystyczne macierzy. Pomyślnie jej utworzenie wymaga dopasowania wartości wszystkich wyspecyfikowanych atrybutów. Specyfikacja składa się ze słowa kluczowego, znaku `=` i wartości. Do podstawowych słów kluczowych, definiujących parametry macierzy należą:

- **uuid** – wartością jest niepowtarzalny identyfikator urządzenia, którym jest 128-mio bitowa liczba zapisana w układzie szesnastkowym. Aby specyfikowana macierz została zainicjalizowana, musi być zgodny z identyfikatorem przechowywanym w bloku identyfikacyjnym (superbloku) macierzy.
- **name** – słowo to umożliwia zdefiniowanie nazwy macierzy, która może zostać wykorzystana przez program *mdadm* podczas inicjalizacji macierzy. Podobnie jak w przypadku

numeru identyfikacyjnego, dla uruchomienia macierzy wymagana jest zgodność nazwy podanej w pliku konfiguracyjnym i zapisanej w bloku identyfikacyjnym. Ponieważ nie wszystkie formaty bloków identyfikacyjnych przechowują nazwę, zaleca się stosowanie numeru identyfikacyjnego.

- **super-minor** – wartością jest liczba całkowita, przechowywana w bloku identyfikacyjnym, zapisana w trakcie jej tworzenia. Jeśli jest to urządzenie `/dev/mdX`, to w momencie tworzenia macierzy do jej bloku identyfikacyjnego, jako wartość *super-minor*, zostanie zapisana wartość *X*.
  - **devices** – słowo umożliwia wyspecyfikowanie urządzeń tworzących macierz. Lista składa się z nazw reprezentujących je plików urządzeń oddzielonych przecinkiem. Każde z występujących urządzeń musi znajdować się w linii *DEVICE*.
  - **level** – wartość określa poziom macierzy RAID, który jest wykorzystywany do jej identyfikacji. Może zostać wpisany ręcznie, jednak zaleca się wykorzystać wynik komendy: `mdadm --examine --scan`. W ten sposób utworzono linię *ARRAY* w przykładowym pliku konfiguracyjnym przedstawionym poniżej.
  - **num-devices** – definiuje liczbę urządzeń w kompletnej, działającej macierzy.
  - **spares** – wartością opcji jest liczba niewykorzystanych (nadmiarowych, zbędnych) urządzeń w jakie została wyposażona macierz. Urządzenie takie może pojawić się przejściowo, jeśli z kompletnej macierzy usuniemy dowolne, a następnie dodamy nowe, to w czasie rekonstrukcji macierzy będzie ono zaznaczone jako zbędne. Przykład ten pokazuje ostatni opis stanu macierzy.
  - **spare-group** – wartość opcji, to nazwa grupy macierzy dyskowych. Zbędne urządzenia mogą w razie potrzeby być przenoszone pomiędzy macierzami dyskowymi, jeśli należą one do tej samej grupy.
  - **auto** – opcja ta stanowi informację dla programu *mdadm* mówiącą, że ma on utworzyć plik urządzenia jeśli on nie istnieje lub jeśli istnieje, ale numer urządzenia nie jest poprawny. Możliwe wartości opcji to *yes* lub *md* dla zwykłych macierzy oraz *mdp*, *part* lub *partition* dla macierzy, które mogą zostać podzielone na partycje (implementuje je jądro systemu Linux od wersji 2.6).
  - **bitmap** – wartością jest ścieżka dostępu do pliku, w którym została zapisana mapa bitowa macierzy. Mapa bitowa jest wykorzystywana do przechowywania informacji o tym, które obszary urządzeń lub partycji składowych macierzy zostały zmodyfikowane od ostatniej ich synchronizacji. Zapis dokonywany jest przez sterownik macierzy, w ściśle określonych momentach czasowych. Informacja ta jest wykorzystywana np. podczas synchronizacji po zaniku zasilania, która przeprowadzona jest jedynie dla obszarów zapisanych w pliku bitmapy.  
Wewnętrzny plik bitmapy, dla macierzy reprezentowanej plikiem urządzenia `/dev/mdX`, tworzymy poleceniem: `mdadm /dev/mdX -Gb internal`, zaś usuwamy: `mdadm /dev/mdX -Gb none`.
  - **metadata** – pozwala zdefiniować format metadanych macierzy. Dostępne wartości to: 0.9, 1, 1.0, 1.1 oraz 1.2. Wartość domyślna to 0.9. Jest ona jednocześnie zalecana ze względu na kompatybilność.
- **MAILADDR** – linia z tym słowem kluczowym służy zdefiniowaniu jednego adresu, na który zostanie wysłana informacja o zajściu zdarzenia wykrytego przez program *mdadm* uruchomiony z opcją **—monitor**.

- **MAILFROM** – słowo kluczowe pozwala na dodanie do listu wysyłanego w przypadku wystąpienia zdarzenia w jakiejś macierzy informacji o nadawcy. Najczęściej ma postać nazwy domeny. Jeśli poszczególne słowa są oddzielone spacjami to są one łączone w postać nazwy symbolicznej. W linii musi pojawić się co najmniej 5 znaków (oprócz znaków białych), aby była ona respektowana.
- **PROGRAM** – linia, która rozpoczyna się tym słowem kluczowym zawiera ścieżkę dostępu (najlepiej bezwzględną) do programu, który zostanie uruchomiony jeśli program *mdadm* z przełącznikiem **—monitor** wykryje potencjalnie istotne dla monitorowanej macierzy zdarzenie. Program zostanie uruchomiony z dwoma lub trzema argumentami, którymi będą: identyfikator zdarzenia, nazwa pliku reprezentującego macierz oraz ewentualnie nazwę urządzenia składowego. Przykład skryptu, który zapewnia podstawową obsługę macierzy w tym zakresie zamieszczono w punkcie 5.4.5. W pliku konfiguracyjnym może znajdować się maksymalnie jedna linia rozpoczynająca się słowem kluczowym **PROGRAM**.
- **CREATE** – linia ta jest wykorzystywana do definiowania wartości domyślnych atrybutów, wykorzystywanych przy tworzeniu macierzy lub definiowaniu jej urządzeń składowych. Do opcji tych należą:
  - **owner**= – wartością opcji jest nazwa lub numer identyfikacyjny użytkownika, który będzie właścicielem indywidualnym zamiast domyślnego (*root*).
  - **group**= – wartością opcji jest nazwa lub numer identyfikacyjny grupy, która będzie właścicielem grupowym zamiast domyślnej (*wheel* lub *disk*).
  - **mode**= – definiuje prawa dostępu do pliku w zapisie ósemkowym.
  - **auto**= – opcja przyjmuje wartości **yes**, **md**, **mdp**, **part**, po których zazwyczaj pojawia się liczba. Stanowi odpowiednik opcji **—auto** programu *mdadm* i definiuje sposób, w jaki będą tworzone wpisy brakujących urządzeń.
  - **metadata**= – umożliwia zdefiniowanie formatu metadanych.
  - **symlinks**=**no** – bez użycia tej opcji, program *mdadm* tworząc plik urządzenia *X* w katalogu */dev/md* tworzy do niego dowiązanie symboliczne o nazwie */dev/mdX*. Użycie opcji spowoduje, że dowiązanie nie zostanie utworzone.

Przykład pliku konfiguracyjnego przedstawiono poniżej:

```

1 #Plik testowy
2 DEVICE /dev/sda7 /dev/sdb2 /dev/sdc2
3 # /dev/md0 robimy przez UUID
4 ARRAY /dev/md0 level=raid5 num-devices=3 UUID=e82e918e:8538d27f:7b12819d:fbd36ea6
5
6 MAILADDR boryczko@agh.edu.pl
7 PROGRAM /usr/sbin/handle-mdadm-events
8 CREATE group=system mode=0640 auto=part-8

```

Przedstawione dotychczas operacje są najczęściej wykonywanymi na macierzach dyskowych. Lista możliwych do wykonania z wykorzystaniem polecenia *mdadm* jest znacznie dłuższa. Przyglądnijmy się zatem podstawowym opcjom komendy. Zaczniemy od trybów pracy, które mogą być następujące:



- *Assemble*, **-A**, **-assemble** – komenda *mdadm* w tym trybie jest wykorzystywana do zbudowania uprzednio zdefiniowanej macierzy. Składowe urządzenia mogą zostać podane jawnie lub odszukane w systemie przez komendę.
- *Build*, **-B**, **-build** – w tym trybie tworzona jest macierz dyskowa, która nie posiada bloku identyfikacyjnego. W przypadku tego typu macierzy program *mdadm* nie rozróżnia czy jest to budowa nowej macierzy, czy budowanie już zdefiniowanej macierzy.
- *Create*, **-C**, **-create** – tryb ten tworzy macierz z blokiem identyfikacyjnym na każdym urządzeniu.
- *Follow or Monitor*, **-F**, **-follow**, **-monitor** – tryb ten umożliwia śledzenie stanu urządzeń wchodzących w skład macierzy oraz zapisuje jego zmiany. Jest dostępny dla macierzy pracujących na poziomie 1, 4, 5, 6, 10 oraz macierzy wieloscieżkowych. Dla macierzy poziom 0 oraz liniowych, które nie mogą mieć urządzeń uszkodzonych, zbędnych lub brakujących tryb ten nie jest wykorzystywany.
- *Grow*, **-G**, **-grow** – opcja umożliwia dodawanie nowych lub usuwanie istniejących urządzeń do macierzy dyskowej.
- *Incremental Assembly*, **-I**, **-incremental** – w trybie tym możliwe jest dodanie jednego urządzenia do konkretnej, uprzednio utworzonej macierzy. Jeśli operacja ta spowoduje, że macierz staje się gotowa do uruchomienia, zostanie ona uruchomiona.
- *Manage* – tryb ten umożliwia działania na poziomie urządzeń składowych macierzy, jak np. usunięcie z macierzy urządzenia uszkodzonego lub dodanie do niej nadmiarowego.
- *Misc* – w tym trybie wykonywane są pozostałe operacje na macierzach dyskowych i ich urządzeniach składowych, jak np. usuwanie bloków identyfikacyjnych czy zbieranie informacji zapisanych o dyskach i partycjach w celu utworzenia macierzy.
- *Auto-Detect*, **-auto-detect** – tryb ten nie działa na wskazanej macierzy, czy urządzeniu składowym. Jest wykorzystywany przez jądro systemu operacyjnego do aktywowania autometrycznie znalezionej macierzy dyskowej np. podczas inicjalizacji systemu.

Opcje wykorzystywane przez komendę *mdadm* można podzielić na właściwe dla odpowiednich trybów pracy oraz ogólne. W trybie tworzenia (*Create*, *Build*) oraz dodawania i usuwania urządzeń składowych (*Grow*) składnia komendy jest następująca:

```
mdadm -build plik_urzadzenia_md -chunk=X -level=Y -raid-devices=Z pliki_urzadzen_skladowych
mdadm -create plik_urzadzenia_md -chunk=X -level=Y -raid-devices=Z pliki_urzadzen_skladowych
```

Do najczęściej wykorzystywanych opcji w tych trybach należą:

- **-n**, **-raid-devices=** – opcja pozwala na wyspecyfikowanie liczby urządzeń aktywnych tworzących macierz. Powiększona o liczbę urządzeń nadmiarowych musi być równa liczbie urządzeń tworzących macierz.
- **-x**, **-sparse-devices=** – opcja służy zdefiniowaniu liczby urządzeń dodatkowych (*eXtra*) w tworzonej macierzy.
- **-z**, **-size=** – wartością opcji jest rozmiar podany w kB, zajmowany przez macierz na poszczególnych jej urządzeniach. Opcja jest dostępna dla poziomów 1, 4, 5, 6. Podany rozmiar musi stanowić wielokrotność rozmiaru obszaru oraz uwzględniać 128 kB rozmiaru z przeznaczeniem na blok identyfikacyjny.

- **-c, -chunk** – opcja umożliwia zdefiniowanie rozmiaru obszaru. Jednostką jest kB. Wartość domyślna to 64 kB.
- **-l, -level** – poziom tworzonej macierzy RAID oznacza wartość opcji, którą może być: linear, raid0, 0, stripe, raid1, 1, mirror, raid4, 4, raid5, 5, raid6, 6, raid10, 10, multipath, mp, faulty.
- **-p, -layout** – opcja może zostać wykorzystana w momencie tworzenia macierzy RAID poziomu 5 i 10 dla zdefiniowania wartości niektórych ich atrybutów. Przykładowo, dla poziomu 5 układ bloku parzystości definiują następujące wartości opcji: left-asymmetric, left-symmetric, right-asymmetric, right-symmetric, la, ra, ls, rs. Wartością domyślną jest left-symmetric.
- **-b, -bitmap** – wartością opcji jest ścieżka dostępu do pliku zawierającego mapę bitową macierzy. Plik nie powinien istnieć. W przeciwnym przypadku należy użyć opcji **-force**. Jeśli jako wartość opcji, zamiast ścieżki dostępu, pojawi się słowo *internal*, plik bitmapy zostanie zapisany w metadanych macierzy.
- **-bitmap-chunk=** – pozwala zdefiniować rozmiar kawałka przestrzeni adresowej macierzy, którego stan będzie określał pojedynczy bit mapy. Jeśli mapa bitowa jest zapisana w pliku, to przyjmuje się, że jeden bit będzie opisywał obszar co najmniej 4 kB, zaś liczba kawałków nie będzie większa niż  $2^{21}$ . W przypadku mapy bitowej umieszczonej w metadanych macierzy, wiadomo jedynie tyle, że rozmiar kawałka jest dobierany tak, aby zapewnić optymalne wykorzystanie przestrzeni adresowej macierzy.
- **-assume-clean** – opcja ta sugeruje komendzie *mdadm*, że macierz istnieje i jest gotowa do użycia. Opcję stosuje się do odtwarzania macierzy dowolnego poziomu po awarii oraz podczas tworzenia macierzy poziomu 1 i 10 w celu wymuszenia resynchronizacji.
- **-backup-file=** – wartością opcji jest ścieżka dostępu do pliku, który ma przechować zawartość macierzy. Jest ona wykorzystywana wraz z opcją **-growth** podczas zwiększania liczby urządzeń macierzy pracującej na poziomie 5, jeśli macierz nie posiada urządzeń nadmiarowych. Plik powinien znajdować się na urządzeniu, które nie tworzy macierzy.
- **-N, -name=** – opcja pozwala zdefiniować nazwę macierzy. Jest to możliwe, jeśli macierz wykorzystuje blok identyfikacyjny w wersji 1. Nazwa może zostać wykorzystana do identyfikowania urządzeń składowych macierzy podczas jej odbudowy.
- **-R, -run** – wymusza uczynienie macierzy aktywnej nawet jeśli jej urządzenia są w danej chwili aktywnie wykorzystane w innych macierzach dyskowych lub systemach plików. Użycie opcji powoduje dodatkowo, iż komenda *mdadm* nie wymaga potwierdzenia wykorzystania takich urządzeń w uruchamianej macierzy.
- **-f, -force** – komenda *mdadm* uruchomiona z tą opcją umożliwia „siłowe” utworzenie macierzy nawet wówczas, jeśli podane wartości atrybutów nie spełniają narzuconych na nie wymagań.
- **-a, -auto{=no,yes,md,mdp,part,p}{NN}** – opcja powoduje utworzenie przez komendę *mdadm* pliku urządzenia reprezentującego tworzoną macierz. Zostanie mu nadany niewykorzystywany, mały (*minor*) numer urządzenia. Jeśli w linii komend lub pliku konfiguracyjnym nie pojawiła się opcja **-auto=**, zostaje domyślnie przyjęte ustawienie **-auto=yes**. Uruchomienie komendy z opcją **-scan** powoduje, że wpisy z pliku konfiguracyjnego dotyczące opcji **-auto** są ignorowane i obowiązują te z linii komend. W przypadku macierzy

partycjonowanych, utworzony zostanie plik urządzenia reprezentującego całą macierz oraz pliki dla domyślnie czterech pierwszych partycji. Zmianę liczby plików dla partycji dokonuje się dopisując wymaganą wartość do wartości opcji, np **-auto=p7**. Jeśli nazwa pliku urządzenia kończy się cyfrą, dodawan jest litera p, a po niej numer partycji, np */dev/wspolny1p4*. W przeciwnym przypadku, do nazwy pliku doklejany jest wprost numer partycji.

- **-symlink=no** – użycie opcji **-a** skutkuje utworzeniem pliku urządzenia reprezentującego macierz w katalogu */dev/md/*, jak również dowiązania symbolicznego w katalogu */dev/* o nazwie rozpoczynającej się od *md* lub *md\_*. Użycie opcji z wartością *no* spowoduje, że dowiązanie nie zostanie utworzone. Opcji z wartością *yes* używa się, jeśli dowiązanie ma zostać utworzone, a w pliku konfiguracyjnym *mdadm.conf* jest to zablokowane.

Składnia komendy *mdadm* użytej dla zbudowania uprzednio zdefiniowanej macierzy może mieć jedną z poniższych postaci ogólnych:

```
mdadm -assemble plik_urzadzenia_md opje_i_pliki_urzadzen_skladowych
```

```
mdadm -assemble -scan opje_i_pliki_urzadzen_skladowych
```

```
mdadm -assemble -scan opcje
```

Spośród opcji, które mogą zostać użyte w tym trybie należy wymienić:

- **-u**, **-uuid=** – opcja służy wyspecyfikowaniu identyfikatora macierzy, która ma zostać utworzona. Urządzenia, które nie są oznaczone wskazanym identyfikatorem nie zostaną wykorzystane do budowy macierzy.
- **-m**, **-super-minor=** – macierz może zostać utworzona również w oparciu o numer (*minor*) urządzeń ją tworzących. Opcja umożliwia podanie numeru urządzeń, która mają ją tworzyć. Urządzenie znajdujące się w danym systemie komputerowym, a posiadające zapisany w bloku identyfikacyjnym numer różny od podanego jako wartość opcji nie zostaną wykorzystane do budowy macierzy.
- **-N**, **-name=** – opcja pozwala podać nazwę macierzy, która ma zostać zbudowana. Nazwa musi być identyczna z tą, jaka została nadana podczas tworzenia macierzy.
- **-f**, **-force** – umożliwia utworzenie macierzy, jeśli bloki identyfikacyjne niektórych urządzeń nie zawierają odpowiednich danych.
- **-R**, **-run** – opcja umożliwia uruchomienie macierzy jeśli obecnie dostępnych jest mniej urządzeń ją tworzących, niż podczas poprzedniego stanu jej aktywności. Uruchomienie komendy *mdadm* bez tej opcji oraz bez opcji **-scan** powoduje, że jeśli nie wszystkie spośród wchodzących w jej skład urządzeń zostaną znalezione, to macierz zostanie zbudowana ale nie będzie uruchomiona.
- **-no-degraded** – działa odwrotnie do opcji **-run** uniemożliwiając uruchomienie macierzy dopóty, dopóki wszystkie tworzące ją urządzenia nie będą dostępne. Opcja ta jest wykorzystywana wraz z opcją **-scan** zazwyczaj, jeśli połączenia z urządzeniami nie są odpowiednio niezawodne.
- **-a**, **-auto{=no,yes,md,mdp,part}** – działanie opcji jest identyczne, jak w trybie tworzenia macierzy.
- **-b**, **-bitmap=** – opcja pozwala wyspecyfikować ścieżkę dostępu do pliku bitmapy powstałego podczas tworzenia macierzy. Jeśli macierz posiada plik bitmapy zapisany w jej metadanych (wewnętrzny), to nie ma potrzeby wykorzystywania tej opcji.

- **–backup-file=** – jeśli podczas zwiększania liczby urządzeń macierzy pracującej na poziomie 5 został wskazany plik kopii zapasowej, a sam proces został przerwany w trakcie budowania macierzy, istnieje możliwość odzyskania danych ze wskazanego pliku. Wartością opcji jest ścieżka dostępu do pliku kopii zapasowej.
- **–U, –update=** – opcja umożliwia zmianę zawartości bloku identyfikacyjnego, zapisanego na każdym urządzeniu wchodzącym w skład macierzy. Wartości opcji wskazują na atrybuty macierzy, które mają zostać zmienione i mogą być następujące:
  - *sparc2.2* – opcja pozwala przekonwertować blok identyfikacyjny macierzy, która została utworzona w systemie pracującym z jądrem w wersji 2.2 na procesorach Sparc.
  - *summaries* – opcja służy weryfikacji ogólnych danych o macierzy, do których należą liczby: wszystkich, pracujących, aktywnych, uszkodzonych i nadmiarowych urządzeń.
  - *uuid* – dla zmiany identyfikator macierzy dyskowej.
  - *name* – dla zmiany nazwy macierzy.
  - *homehost* – dla zmiany nazwy hosta, na którym fizycznie znajduje się macierz.
  - *resync* – ustawia flagę w bloku identyfikacyjnym, wskazującą na konieczność synchronizacji macierzy.
  - *byteorder* – ustawia flagę umożliwiającą przenoszenie macierzy między systemami komputerowymi różniącymi się reprezentacją maszynową liczb. Wartość dostępna w wersji 0.9 bloku identyfikacyjnego.
  - *devicesize* – opcja dostępna jest w wersjach 1.1 i 1.2 bloku identyfikacyjnego i wykorzystywana w przypadku zmiany rozmiaru jednego z urządzeń składowych (zazwyczaj zwiększenia). Sama zmiana rozmiaru urządzenia, w szczególności jego zwiększenie, nie spowoduje, że będzie ona w całości dostępna w macierzy. Konieczne jest wprowadzenie zmian w stosownych metadanych, co umożliwia ta właśnie opcja.
  - *super-minor* – umożliwia zmianę wartości będącą preferowanym numerem *minor* macierzy w blokach identyfikacyjnych zapisanych na wszystkich jej urządzeniach. Opcję stosuje się, jeśli uruchomienie komendy z opcją **–examine** zwraca różne numery zapisane w blokach identyfikacyjnych na poszczególnych urządzeniach. W niektórych przypadkach zmiana wartości pola dokonywana jest automatycznie przez sterownik macierzy znajdujący się w jądrze systemu. W szczególności ma to miejsce, podczas pierwszego zapisu do macierzy poziomu zapewniającego nadmiarowość danych, w systemie wykorzystującym jądro w wersji 2.6.
- **–auto-update-homehost** – opcja ta jest wykorzystywana w przypadku automatycznego budowania macierzy. W takim przypadku, jeśli dla bieżącego hosta nie znaleziono odpowiadającej mu macierzy, program *mdadm* wyszukuje występujące w systemie macierze, a następnie w blokach identyfikacyjnych wpisuje nazwę bieżącego hosta, jako systemu dla niej macierzystego.

Ogólna postać składni komendy *mdadm* w trybie administrowania jest następująca: *mdadm plik\_urządzenia\_md opcje pliki\_urządzeń\_skladowych*

W trybie tym, do zarządzania macierzą wykorzystuje się następujące opcje:

- **–a, –add** – pozwala dodać wskazane urządzenie do pracującej macierzy dyskowej.

- **-re-add** – opcja służy dodaniu urządzenia, które z macierzy zostało usunięte.
- **-r, -remove** – opcja umożliwia usunięcie z macierzy wskazanego urządzenia. Usuwane urządzenie nie może być aktywne, może być zatem nadmiarowym lub zaznaczonym jako uszkodzone.
- **-f, -fail** – zaznacza wskazane urządzenie jako uszkodzone. Jako wartość opcji, zamiast ścieżki dostępu do pliku urządzenia może pojawić się słowo *detached*.

Spośród innych, użytecznych opcji wymienić należy:

- **-Q, -query** – komenda *mdadm* użyta z tą opcją wymaga podania jako argumentu ścieżki dostępu do pliku urządzenia. Po pierwsze sprawdzane jest, czy plik reprezentuje macierz dyskową. Jeśli tak, to wypisywana jest jej krótka charakterystyka w postaci informacji o pojemności, poziomie i liczbie urządzeń. Jeśli dany plik nie reprezentuje macierzy, to komenda sprawdza, czy reprezentowane przez niego urządzenie jest składnikiem macierzy. Jeśli tak, to pojawia się informacja w postaci numeru urządzenia w macierzy, poziomie macierzy oraz nazwy reprezentującego ją pliku. W przeciwnym przypadku pojawia się informacja, że wskazany plik nie reprezentuje macierzy. W naszym systemie działanie opcji jest następujące:

```

1 [root@messy ~]# mdadm -q /dev/md0
2 /dev/md0: 1035.25MiB raid5 3 devices, 0 spares. Use mdadm --detail for more detail.
3 [root@messy ~]# mdadm -q /dev/sdc2
4 /dev/sdc2: is not an md array
5 /dev/sdc2: device 2 in 3 device active raid5 /dev/md0. Use mdadm --examine for more detail.
6 [root@messy ~]# mdadm -q /dev/sda2
7 /dev/sda2: is not an md array

```

- **-D, -detail** – opcja pozwala wypisać charakterystykę macierzy dyskowych, których reprezentujące pliki pojawiły się jako argumenty komendy.
- **-Y, -export** – jeśli komendę *mdadm* uruchomiono z opcją **-D**, to informacje o macierzy wypisywane będą w postaci *nazwa\_ atrybutu=wartość*:

```

1 [root@messy ~]# mdadm --export --detail /dev/md0
2 MD_LEVEL=raid5
3 MD_DEVICES=3
4 MD_METADATA=0.90
5 MD_UUID=97f1e3e1:f2304505:59cb159f:4a572848

```

- **-E, -examine** – jako argument podaje się nazwę pliku urządzenia wchodzącego w skład macierzy. W wyniku działania wypisana zostaje informacja zapisana w bloku identyfikacyjnym zapisanym na tym urządzeniu.
- **-sparc2.2** – opcja jest wykorzystywana do zmiany formatu bloku identyfikacyjnego, zapisanego na urządzeniu, którego plik podany został jako argument. Wykorzystuje się ją jeśli macierz została utworzona w systemie z łańcuchym jądrem w wersji 2.2. Należy użyć również opcji **-examine**.

- **-X, -examine-bitmap** – opcja pozwala wypisać informacje o pliku zawierającego mapę bitową macierzy. Argumentem jest nazwa pliku reprezentującego macierz w przypadku bitmapy zapisanej w metadanych systemu plików, lub ścieżka dostępu do pliku, jeśli zapisana jest ona poza macierzą:

```

1 [root@messy ~]# mdadm -X /tmp/bitmap
2     Filename : /tmp/bitmap
3     Magic : 6d746962
4     Version : 4
5     UUID : e1e3f197:054530f2:9f15cb59:4828574a
6     Events : 4
7     Events Cleared : 0
8     State : OK
9     Chunksize : 4 KB
10    Daemon : 5s flush period
11    Write Mode : Normal
12    Sync Size : 530048 (517.71 MiB 542.77 MB)
13    Bitmap : 132512 bits (chunks), 0 dirty (0.0%)

```

- **-R, -run** – umożliwia uruchomienie macierzy zbudowanej nie ze wszystkich urządzeń ją tworzących.
- **-S, -stop** – deaktywuje macierz zwalniając tworzące ją urządzenia. Nie usuwa wpisów z pliku konfiguracyjnego */etc/mdadm.conf*.
- **-o, -readonly** – działanie opcji polega na udostępnieniu macierzy w trybie tylko do odczytu. Aby wykonanie komendy zakończyło się sukcesem, system plików utworzony w danej macierzy nie może być zamontowany. Zamontowanie systemu plików z macierzy będącej w trybie tylko do odczytu spowoduje, że system plików będzie dostępny tylko do odczytu.
- **-w, -readwrite** – opcja pozwala udostępnić macierz w trybie do odczytu i zapisu. Przełączenie do tego trybu jest możliwe z zamontowanym systemem plików utworzonym w danej macierzy. Jeśli macierz była w trybie tylko do odczytu, a system plików był zamontowany, to pomimo zmiany trybu pracy macierzy, będzie on nadal dostępny tylko do odczytu – należy zamontować go ponownie w trybie do odczytu i zapisu (np. komendą `mount -o remount,rw /dev/md0`).
- **-zero-superblok** – opcja pozwala zapisać blok identyfikacyjny na wskazanym urządzeniu składowym macierzy bajtami o wartości 0. Stosowana jeśli blok uległ uszkodzeniu i jego zawartość jest niepoprawna.
- **-W, -wait** – operacje synchronizacji zawartości urządzeń są czasochłonne. Użycie tej opcji spowoduje, że komenda *mdadm* zakończy się po zakończeniu tego procesu.

W trybie przyrostowego budowania (*Incremental Assembly*) macierzy możliwe są następujące postacie składni:

```
mdadm -incremental [-run] [-quiet] pliki_urządzeń_składowych
```

```
mdadm -incremental -rebuild
```

*mdadm* **-incremental -run -scan**

Opcje możliwe do wykorzystania w tym trybie są następujące:

- **-rebuild-map, -r** – opcja powoduje uaktualnienie pliku konfiguracyjnego */var/run/mdadm/map*, wykorzystywanego przez program *mdadm* w trakcie procesu budowania macierzy.
- **-run, -R** – opcja umożliwia rozpoczęcie procesu budowy macierzy jeśli tylko numer minimalny urządzenia staje się dostępny. Jeśli opcja nie zostanie użyta, to kryterium rozpoczęcia budowy jest dostępność urządzeń tworzących macierz.
- **-scan, -s** – opcja ma znaczenie przy użyciu z opcją **-R** i powoduje analizę pliku mapy dla macierzy dyskowej, która właśnie jest budowana w sposób przyrostowy w celu jej uruchomienia, jeśli to jeszcze nie nastąpiło.

Składnia uruchomienia komendy *mdadm* w przypadku trybu monitorowania jest następująca:  
*mdadm* **-monitor** *opcje urządzenia*

Spośród grupy opcji wykorzystywanych do monitorowania macierzy dyskowych, do najczęściej wykorzystywanych należą:

- **-m, -mail** – opcja wymaga podania adresu e-mail, na który zostanie wysłana informacja w przypadku stwierdzenia nieprawidłowości w działaniu urządzeń macierzy.
- **-p, -program, -alert** – argumentem dla opcji jest ścieżka dostępu do pliku programu, który ma zostać uruchomiony w momencie stwierdzenia nieprawidłowości w działaniu macierzy. Najczęściej jest to skrypt, który można napisać we własnym zakresie. Przykładowy przedstawiono w punkcie 5.4.5.
- **-y, -syslog** – użycie opcji powoduje, że do raportowania wykorzystany zostanie systemowy mechanizm dziennikowania, omówiony w punkcie 4.4.1.
- **-d, -delay** – wartością jest liczba sekund co jaką badany jest stan macierzy. Wartością domyślną jest 60.
- **-f, -daemonise** – użycie opcji spowoduje uruchomienie programu *mdadm* jako demon, w tle. W praktyce skutkuje to wykonaniem funkcji systemowej *fork* i utworzeniem procesów potomnych, których numery identyfikacyjne zostaną wypisane na standardowym wyjściu. Opcja stosowana jest z opcją **-scan**. Dodatkowo, w pliku konfiguracyjnym powinien być podany adres e-mail, na który ma zostać wysłana informacja oraz program wywoływany w przypadku wykrycia nieprawidłowości w działaniu macierzy.
- **-i, -pid-file** – wartością opcji jest ścieżka dostępu do pliku, do którego zapisane zostaną numery identyfikacyjne procesów jeśli program *mdadm* został uruchomiony jako demon z opcją **-f**.
- **-1, -one-shot** – powoduje jednokrotne monitorowanie stanu macierzy. Generuje zdarzenie *NewArray* oraz istotniejsze *DegradedArray* i *SparseMissing*. Zalecane jest uruchamianie polecenia: *mdadm -monitor -scan -1* z wykorzystaniem demona zegarowego (*crontab*), co pozwoli śledzić stan macierzy.
- **-t, -test** – komenda użyta z tą opcją generuje alarm o nazwie *TestMessage*, jeśli w systemie zostanie utworzona nowa macierz (program *mdadm* uruchomiony z tą opcją: *mdadm -monitor -scan -t* śledzi stan macierzy dyskowych w systemie). Skutkuje to wysłaniem informacji na zdefiniowany w pliku konfiguracyjnym adres. Stąd najczęstszym zastosowaniem opcji jest testowanie poprawności tego fragmentu konfiguracji.

Program *mdadm* posiada również opcje, które są niezależne od wybranego trybu pracy. Do najczęściej stosowanych należą:

- **-h, -help** – wypisuje krótką informację o składni i podstawowych opcjach.
- **-v, -verbose** – włącza szczegółowy tryb raportowania.
- **-q, -quiet** – użycie tej opcji powoduje wybranie „cichego” trybu raportowania.
- **-b, -brief** – ustawia oszczędniejszy od opcji **-v** tryb raportowania.
- **-f, -force** – umożliwia bardziej „agresywne” działanie komendy *mdadm*, co umożliwia wykonanie niektórych, niebezpiecznych dla przechowywanych w macierzy danych operacji.
- **-c, -config=** – wartością opcji jest ścieżka dostępu do pliku konfiguracyjnego (domyślnie jest to plik */etc/mdadm.conf*, a w przypadku jego braku plik */etc/mdadm/mdadm.conf*). Jeśli zamiast ścieżki dostępu pojawi się napis *partitions*, to komenda *mdadm* sięgnie do pliku */proc/partitions* w celu znalezienia listy urządzeń, z których potencjalnie może zostać utworzona macierz dyskowa. Użycie napisu *none* spowoduje działanie identyczne, jak w przypadku istnienia pustego pliku konfiguracyjnego.
- **-s, -scan** – opcja wymusza przeszukanie pliku konfiguracyjnego lub pliku */proc/mdstat* w celu znalezienia brakujących informacji. Mogą nimi być: nazwy urządzeń składowych, nazwy macierzy dyskowych lub ich numery identyfikacyjne.
- **-e, -metadata=** – opcja jako wartość przyjmuje numer wersji formatu bloku identyfikacyjnego (metadanych macierzy). Wartość domyślna to 0.9, zarówno podczas tworzenia macierzy, jak i innych, wykonywanych na niej operacji. Ustawienie innej wartości domyślnej odbywa się poprzez nadanie odpowiedniej wartości zmiennej *CREATE* w pliku konfiguracyjnym *mdadm.conf*. Obsługiwane obecnie wersje to: 1, 1.0, 1.1, 1.2. Należy zwrócić uwagę na ich niekompatybilność wynikającą choćby z różnego miejsca przechowywania bloku identyfikacyjnego (w wersji 1.0 jest to koniec urządzenia, w wersji 1.1 początek, zaś w wersji 1.2 znajduje się on w odległości 4 kB od początku urządzenia).
- **-homehost=** – wartość opcji zasłania wartość zmiennej *HOMEHOST* nadaną jej w pliku konfiguracyjnym *mdadm.conf*. Zmienna wskazuje na hosta, w którym dana macierz jest zdefiniowana. Nazwa hosta zostaje również zapisana bloku identyfikacyjnym macierzy. W wersji 0.9 jest ona poprzedzona nazwą macierzy, a jej skrót uzyskany funkcją SHA1 stanowi część identyfikatora macierzy.

Na zakończenie krótka informacja o plikach udostępniających informacje o stanie macierzy dyskowej. Jak wiemy, obraz systemu przechowywany jest w wirtualnym systemie plików zamontowanym w katalogu *proc*. W pliku tekstowym o nazwie *mdstat* znajdziemy informację o macierzach dyskowych pracujących w naszym systemie. Zawartość pliku z przykładowego systemu zamieszczono poniżej:

```

1 [root@messy mdadm]# cat /proc/mdstat
2 Personalities : [raid6] [raid5] [raid4]
3 md0 : active raid5 sdc3[3](S) sdc2[4] sdb2[1] sda7[0]
4      1060096 blocks level 5, 64k chunk, algorithm 2 [3/2] [UU_]
5      bitmap: 0/65 pages [0KB], 4KB chunk, file: /tmp/bitmap_md0
6      [=====>.....]   recovery = 40.3% (214048/530048) finish=0.5min speed=9306K/sec

```



```
unused devices: <none>
```

Plik rozpoczyna linia ze słowem *Personalities*. Mówi ona jakie poziomy RAID są aktualnie obsługiwane przez jądro systemu operacyjnego. W dalszej części plik budowę zwrotkową. Każda zwrotka opisuje stan jednej macierzy dyskowej i rozpoczyna się od jej nazwy (linia 3). Opis składa się z dwóch linijek. W pierwszej (linia 3) podany jest stan macierzy (active), poziom (raid5) oraz lista tworzących ją urządzeń w postaci nazw reprezentujących ich plików. Za każdą nazwą, w nawiasach kwadratowych podany jest numer urządzenia w macierzy. Urządzenia aktywne nie są dodatkowo oznaczane. Jeśli urządzenie jest nadmiarowym, to w nawiasach półokrągłych pojawia się litera S (*sparse*). Urządzenie uszkodzone jest oznaczane literą F (*Fail*). Druga linia opisu (linia 4) zawiera informacje o rozmiarze podaną w blokach, poziomie macierzy, rozmiarze fragmentu macierzy (w kB), algorytmie wyliczania sumy kontrolnej oraz stanie urządzeń zapisanym w dwóch parach nawiasów kwadratowych. Pierwsza para mówi o liczbie urządzeń tworzących macierz i liczbie urządzeń aktualnie dostępnych (w naszym przypadku 3 urządzenia tworzące, 2 dostępne). Informacja zapisana w drugiej parze nawiasów pozwala stwierdzić stan kolejnych urządzeń macierzy. Litera U (ang. *Up*) mówi, że urządzenie jest aktywne, \_ , że jest niedostępne. Jeśli macierz dyskowa posiada zewnętrzny plik, w którym przechowywana jest mapa bitowa jej obszarów, to w pliku */proc/mdstat*, w zwrotce opisującej daną macierz, pojawi się linia ze słowem kluczowym *bitmap* (linia 5). Zawiera ona informację o liczbie stron wykorzystanych w pliku i całkowitej liczbie stron dostępnych (0/65), rozmiarze wykorzystanej liczby stron pliku podanym w kB (0kB), rozmiarze pojedynczego kawałka macierzy odpowiadającemu jednemu bitowi w mapie oraz nazwie pliku, w którym mapa jest przechowywana. Linia 6 pojawi się, jeśli macierz znajduje się w stanie synchronizacji lub odtwarzania i mówi o zaawansowaniu tego procesu.

Pliki pomocnicze znajdują się w katalogu */var/run/mdadm*. Są to pliki tekstowe *map* oraz *mdadm.pid*. Zawartość pierwszego z nich można utworzyć komendą *mdadm -I -r*. Przykładowo może wyglądać następująco:

```
md0 0.90 a9326921:38c2b3d1:dd5af398:390b80c4 /unknown
```

Plik ma budowę linikową. Linie rozpoczyna informacja o nazwie pliku urządzenia reprezentującego macierz. Następnie znajduje się numer wersji bloku identyfikacyjnego, niepowtarzalny identyfikator globalny oraz

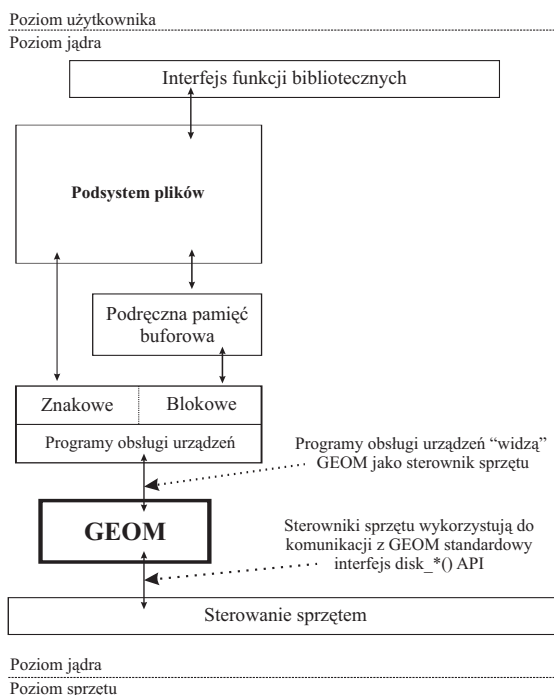
W pliku *mdadm.pid* zapisany jest numer identyfikacyjny procesu monitorującego stan macierzy dyskowych pracujących w systemie.

### 3.7.3 Organizacja pamięci masowej w dystrybucji FreeBSD

#### GEOM w systemie FreeBSD

GEOM jest systemem zarządzania pamięcią masową. Pojawił się już w wersji 5.0, a jego nazwa ma sugerować szeroko uwzględnianą geometrię dysku, co przekłada się bezpośrednio na wydajność znajdującego się na nim systemu plików. System ten zbudowany jest z modułów, z których każdy obsługuje pewną jego funkcjonalność. Moduły te pracują pomiędzy programami obsługi urządzeń, a sterownikami urządzeń. Schematycznie przedstawiono to na rysunku 3.52, będącym modyfikacją ogólnego rysunku 1.2.

Jak w każdym rozwiązaniu tego typu pojawiają się nowe terminy, bez znajomości których zrozumienie idei jego działania, a co za tym idzie administrowanie nim nie jest możliwe. W systemie GEOM do podstawowych należą:



Rysunek 3.52: Moduły GEOM stanowią warstwę pośredniczącą pomiędzy programami obsługi urządzeń, a ich sterownikami.

- transformacja (*transformation*) – unikalny sposób modyfikowania żądań podsystemu wejścia-wyjścia wprowadzony w celu osiągnięcia danej funkcjonalności. Należą do niego: wprowadzenie podziału dysku na partycje (*BSD*, *MBR*, *GPT*, *PC98*, ...), kopia lustrzana, paskowanie, RAID-5, kontrola integralności.
- klasa (*class*) – to implementacja danej transformacji. Do klasy należy MBR odpowiedzialny za dostarczanie informacji o podziale na partycje, *BSD*, kopia lustrzana, ...
- geom (pisane małymi literami) – to instancja, czyli konkretna implementacja danej klasy. Przykładowo jest to MBR, który przechowuje informacje o podziale na partycje, *BSD* dokonujący podziału na partycje urządzenia *ad0s1*, kopia lustrzana zapewniająca synchronizację przykładowo urządzeń *ad2* i *ad3*.
- dostawca (*Provider*) – definiowany jako punkt dostępowy (*service point*) dostarczany przez system GEOM. Można go rozumieć jako plik urządzenia w katalogu */dev* (np. */dev/ad0*, */dev/ad0s1*, */dev/ad0s1a*, */dev/ad0.ad1.mirror*).
- konsument (*consumer*) – punkt montowania, którym geom podłącza się do dostawcy. Nie posiada on nazwy, ale w systemie nie jest anonimowy.

W każdym systemie geom może występować  $0 \dots N$  konsumentów i  $0 \dots M$  dostawców. Każdy konsument może zostać dołączony do pojedynczego dostawcy. Dostawca może posiadać co najmniej jednego konsumenta. Topologia połączeń musi być grafem skierowanym – pętle nie są dopuszczalne.

Modularna budowa sprawia, że system jest systemem otwartym – użytkownicy mogą implementować i dodawać swoje moduły. Omawiany uprzednio system LVM jest zamkniętym. Przyjmując jako kryterium podziału pełnioną funkcję, możemy wyróżnić cztery podstawowe grupy modułów:

1. Moduły odpowiedzialne za sposób przechowywania danych (*Storage modules*):
  - *geom\_stripe* – moduł implementuje macierz RAID 0.
  - *geom\_mirror* – moduł implementuje macierz RAID 1.
  - *geom\_raid3* – moduł implementuje macierz RAID 3.
  - *geom\_raid5* – moduł implementował macierz RAID 5. Obecnie nie stosowany.
  - *geom\_concat* – moduł umożliwia łączenie rozłącznych fragmentów dysku w ciągły obszar (konkatenacja).
  - *geom\_vinum* – to moduł zarządcy przestrzeni dyskowej umożliwiający implementację macierzy RAID poziomów 0, 1, 4 oraz 5.
  - *geom\_ccd* – praktycznie wyparty przez moduł *geom\_vinum*. Umożliwia tworzenie macierzy RAID na poziomie 0 i prymitywnych macierzy RAID 1.
2. Moduły umożliwiające szyfrowanie i kompresję przechowywanych danych (*Encryption and compression modules*):
  - *geom\_eli* – lub GELI (*GEOM ELI*) jest modulem umożliwiającym kodowanie informacji zapisywanej na urządzeniach blokowych. Wprowadzony w wersji 6.0.
  - *geom\_bde* – nazywany również GBDE (*GEOM Based Disk Encryption*), umożliwia wprowadzenie specyficznego sposobu kodowania informacji zapisanej na dysku. Pojawił się w wersji 5.0.
  - *geom\_shsec* – moduł wykorzystywany w bezpiecznej komunikacji, np. z dyskiem sieciowym (*shared secret*).
  - *geom\_uzip* – moduł umożliwia odczyt plików kompresowanych algorytmem ZIP.
3. Moduły systemów plików (*Filesystem modules*):
  - *geom\_label* – umożliwia wprowadzenie nazw (etykiet) dla źródeł (dostawców) przestrzeni dyskowej.
  - *geom\_journal* – wprowadza dziennikowanie dla systemu plików UFS (*Unix File System*).
  - *geom\_cache* – moduł zarządza pamięcią podręczną systemu plików, zwiększając jego efektywność.
4. Moduły wspierające tworzenie i zarządzanie zasobami wirtualnymi (*Virtualization modules*):
  - *geom\_md* – tworzy dysk wirtualny wykorzystując plik, plik wymiany pamięci wirtualnej lub pamięć operacyjną.
  - *geom\_nop* – umożliwia tworzenie modułu wykorzystywanego do debuggowania i testowania.
  - *geom\_gate* – umożliwia tworzenie dysku wirtualnego z wykorzystaniem zasobów sieciowych.

- *geom\_virstor* – moduł ten stosowany jest do tworzenia przestrzeni adresowych o rozmiarach większych, niż dostępne przez dostarczycieli w systemie GEOM.
- *geom\_linux\_lvm* – moduł umożliwia odczyt wolumenów LVM w wersji 2 systemów linuksowych.

W dalszej części przedmiotem naszych zainteresowań będą funkcjonalności udostępniane przez moduł *geom\_vinum* oraz wykorzystywany przez niego moduł *geom\_label*.

### Vinum w systemie FreeBSD

Rozpoznawanie funkcjonalności modułu *geom\_vinum* rozpoczniemy od omówienia stosowanej nomenklatury. W systemie Vinum dyski podzielone zostały na trzy odrębne komponenty. Są to:

1. **Wolumeny** (*volume*) – w nomenklaturze Vinum odpowiadają one dyskom wirtualnym. Funkcjonalnie wolumen przypomina partycję dysku fizycznego, na której można tworzyć system plików, zapisywać dane na partycje oraz dane odczytywać. Dla użytkowników systemu wolumeny będą widoczne jak zwykłe partycje. W obrębie partycji nie można mieszać danych przechowywanych w systemie Vinum i w standardowych systemach plików w obrębie jednej, fizycznej partycji.
2. **Napędy** (*drive*) – pod tą nazwą kryją się partycje, które zostały przydzielone do wykorzystania przez system Vinum. Jeden wolumen może zostać zbudowany z wielu napędów.
3. **Poddyski** (*subdisk*) – to obszary przeznaczone do przechowywania danych przez system Vinum na określonym napędzie.

Reasumując, wolumen składa się z napędów, a napęd z poddysków. Dane w wolumenach przechowywane są w kopiach zwanych pleksami (*plex*). Są one dostępne jeśli pleks jest kompletny, czyli wszystkie dyski dostarczające przestrzeni adresowej działają poprawnie. Tworzenie kopii lustrzanej polega na przechowywaniu więcej niż jednego pleksu w wolumenie. Maksymalna liczba pleksów w wolumenie wynosi 8. Układ pleksów określa rodzaj używanej macierzy RAID. Dostępne są następujące rodzaje pleksów:

- *Pleks połączony* (*concatenated plex*) – umożliwia utworzenie jednego wolumenu zawierającego wszystkie dostępne poddyski, ułożone w zadanej kolejności. Otrzymujemy jeden pleks, zatem dane przechowywane są w jednej kopii. Ten rodzaj pleksów jest wykorzystywany do budowy wolumenów o dużych pojemnościach, przekraczających możliwości pojedynczego dysku fizycznego. Jego zaletą jest możliwość łatwej rozbudowy.
- *Pleks paskowany* (*striped plex*) – funkcjonuje jak macierz RAID 0, rozmieszczając dane w taki sposób, aby zwiększyć wydajność. Poddyski w pleksie paskowanym muszą mieć ten sam rozmiar. Minimalna liczba poddysków wynosi 2.  
Używanie jednego pleksu nie daje nadmiarowości. Można zbudować dwa pleksy paskowane na czterech dyskach i utworzyć z nich macierz RAID 0. Nadmiarowość w takim rozwiązaniu jest jednak większa niż w minimalnej konfiguracji RAID 5.
- *Pleks lustrzany* (*mirrored plex*) – przechowuje kopie danych. Wszystkie poddyski muszą mieć ten sam rozmiar. Kopia lustrzana wymaga użycia co najmniej dwóch pleksów.
- *Pleks RAID 5* – implementuje macierz RAID 5. Wymagane są co najmniej trzy poddyski o tym samym rozmiarze.

Proces tworzenia pleksu składa się z trzech kroków:

1. Utworzenie lub wybranie partycji, które zostaną napędami systemu Vinum.
2. Utworzenie poddysków na wskazanych napędach.
3. Skonfigurowanie Vinum tak, aby utworzył odpowiednie pleksy.

**Przygotowanie napędu Vinum**

**Przydzielanie partycji Vinum**

## Rozdział 4

# Procesy

### 4.1 Wprowadzenie

Proces w systemie Unix jest żywą instancją lub inaczej wykonaniem programu. System operacyjny dostrzega w nim segmenty, do których zaliczamy: segment kodu, segment danych, segment BSS (od angielskiej nazwy *Block Started by Symbol*) oraz segment stosu, dokładniej stosów, gdyż proces może pracować w trybie użytkownika lub trybie jądra, a w każdym z nich wykorzystywany jest oddzielny stos. Segment kodu zawiera kod binarny aktualnie wykonywanego programu. Znajduje się w nim zatem kod zaimplementowanych przez nas funkcji oraz kod funkcji dołączonych z bibliotek lub jego adres umożliwiający dynamiczne załadowanie w razie potrzeby. Zapisane w nim adresy funkcji pozwalają na ich lokalizację. Segment danych zawiera zainicjalizowane zmienne globalne, zdefiniowane w programie. System operacyjny musi bowiem wiedzieć, jakie wartości zostały nadane tym zmiennym. Adres segmentu danych można ustalić na podstawie adresu zmiennej globalnej. Zmienne globalne, które nie są zainicjalizowane trafiają do segmentu BSS. Zmienne lokalne, podobnie jak adresy instrukcji wywołania funkcji wykorzystywane podczas powrotu z ich wykonania, zajmują miejsce na stosie.

Rozważmy prosty program napisany w języku C, którego kod źródłowy zamieszczono poniżej:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <unistd.h>
5
6  int global_init_data = 3;
7  int global_no_init_data;
8
9  int print_adres_info(void) {
10
11     int local_data = 1;
12     printf ("Numer identyfikacyjny procesu: %d\n", getpid());
13     printf ("Adresy nalezace do blokow:\n");
14     printf ("kodu: %p\n", &print_adres_info);
15     printf ("danych: %p\n", &global_init_data);
16     printf ("BSS: %p\n", &global_no_init_data);
17     printf ("stosu: %p\n", &local_data);
```

```
18     return 0;
19 }
20
21 int main () {
22     print_adres_info ();
23     return 0;
24 }
```

Zmienna *global\_init\_data* (linia 6) zostanie umieszczona w segmencie danych, zaś zmienna *global\_no\_init\_data* (linia 7) w segmencie BSS. Zmienna *local\_data* (linia 11) trafi do segmentu stosu. Adres początku instrukcji funkcji *print\_adres\_info* należy do segmentu kodu. Adres instrukcji wywołania tej funkcji (linia 22) pojawi się na stosie trybu użytkownika w momencie jej wywołania.

Wykonanie powyższego programu dało następujący wynik:

```
1 Numer identyfikacyjny procesu: 24585
2 Adresy należące do bloków:
3 kodu: 0x80483a0
4 danych: 0x80496b0
5 BSS: 0x80496b8
6 stosu: 0xfef42ff4
```

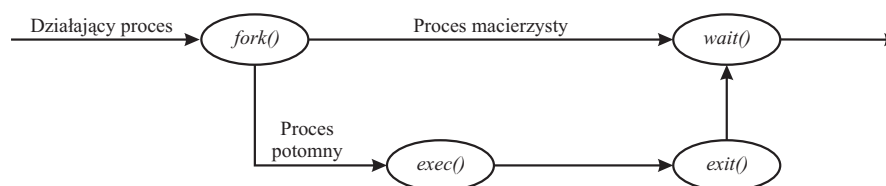
Kompilacja kodu źródłowego utworzy plik wykonywalny, który będzie składał się z kilku części. Należą do nich:

- Zbiór nagłówków opisujących atrybuty pliku.
- Instrukcje (kod) programu.
- Reprezentacja w języku maszynowym zmiennych globalnych mających przypisane wartości początkowe oraz informacje o ilości miejsca koniecznej do przechowania zmiennych globalnych niezainicjalizowanych (rozmiar segmentu BSS), którym jądro w momencie rozpoczynania wykonywania procesu nadaje wartość 0.
- Tablica symboli zawierająca adresy zmiennych i funkcji.

Format pliku wykonywalnego może być różny. Najczęściej spotykane to *a.out* zastępowany przez *ELF* (od *Executable and Linking Format*). Podstawowa różnica dotyczy bibliotek współdzielonych, których tworzenie w wielu przypadkach może być uciążliwe (np. dla systemu X Windows zajmującego duże drzewo katalogów). Dodatkowo format *a.out* nie pozwala na użycie funkcji systemowej *dlopen* wywołującej program dynamicznie ładujący dodatkowe biblioteki współdzielone.

W systemie Unix istnieje jeden mechanizm tworzenia procesu, wykorzystujący funkcję systemową *fork* (rysunek 4.1). Każdy proces, za wyjątkiem procesu o numerze 0, jest tworzony przez wykonanie przez inny proces funkcji *fork*. Proces ją wykonujący nazywa się procesem *macierzystym*, a proces nowoutworzony procesem *potomnym*. Stąd każdy proces ma jeden proces macierzysty, natomiast może mieć wiele procesów potomnych.

Po utworzeniu procesu potomnego jądro ładuje fragmenty pliku wykonywalnego do odpowiednich segmentów procesu potomnego wykorzystując funkcję systemową *exec*. Ścieżka dostępu do pliku zawierającego kod uruchamianego procesu jest jednym z argumentów wywołania funkcji.



Rysunek 4.1: Mechanizm tworzenia procesu w systemach uniksowych.

Uruchomienie nowego procesu wymaga współpracy podsystemu zarządzania procesami z podsystemem pamięci masowej. Ponieważ współpraca z urządzeniami wejścia/wyjścia jest kosztowna czasowo, więc funkcja *exec* najczęściej nie ładuje całego kodu procesu potomnego, lecz jedynie fragment niezbędny do rozpoczęcia jego działania. Dalsze fragmenty są ładowane w zależności od potrzeb. Po uruchomieniu procesu potomnego, proces macierzysty przechodzi do wykonywania funkcji systemowej *wait*, oczekując na zakończenie wykonywania procesu potomnego. Proces potomny kończy swoje działanie wykonując funkcję *exit*. Argumentem funkcji *exit* jest liczba całkowita, niosąca informację o sposobie zakończenia wykonywania procesu. Obowiązuje prosta zasada mówiąca, że jeżeli wykonanie przebiegło bez błędów, to zwracana jest wartość 0, a w przeciwnym przypadku wartość różna od zera, na podstawie której można określić błąd wykonania. Wykonanie przez proces funkcji *exec* skutkuje zwolnieniem przydzielonych mu segmentów.

#### 4.1.1 Struktury opisujące proces w systemie operacyjnym

Jądro systemu operacyjnego przechowuje informacje o uruchomionych procesach w statycznej *tablicy procesów*. Dodatkowo, każdy proces ma przydzielany dynamicznie, w trakcie jego tworzenia tzw. *u-obszar* (rysunek 4.2). Rekord w tablicy procesów zawiera między innymi:

- Pole stanu określające w jakim aktualnie stanie znajduje się proces.
- Pola, które pozwalają jądro na lokalizację segmentów procesu oraz jego *u-obszaru*, gdy ten znajduje się w pamięci operacyjnej, jak i w pliku wymiany. Informacja ta jest wykorzystywana podczas przełączania kontekstu do procesu wówczas, gdy przechodzi on ze stanu „gotowy do wykonania w pamięci” do stanu „wykonywany w trybie jądra” lub ze stanu „wywłaszczony” do stanu „wykonywany w trybie użytkownika”. Jest ona również wykorzystywana podczas przenoszenia procesu do pliku wymiany. Pozycja w tablicy procesów zawiera także informację o aktualnym rozmiarze procesu, która informuje jądro o tym, ile pamięci należy zarezerwować dla danego procesu.
- Identyfikatory właścicieli procesu. Pozwalają one określić prawa procesu w systemie.
- Numer identyfikacyjny procesu i jego procesu macierzystego, określające zależności między procesami.
- Deskryptor zdarzenia, jeśli proces znajduje się w stanie uśpiony.
- Parametry dla systemu szeregowania zadań do procesora.
- Pola sygnałów zawierające informację o sygnałach wysłanych do procesu, ale jeszcze nie obsłużonych.



- Liczniki czasów wykonania procesu i wykorzystania zasobów systemu. Są one wykorzystywane między innymi przez podsystem kolejkowania zadań do wyznaczania wartości priorytetu wskazującego na miejsce procesu w kolejce zadań do procesora.

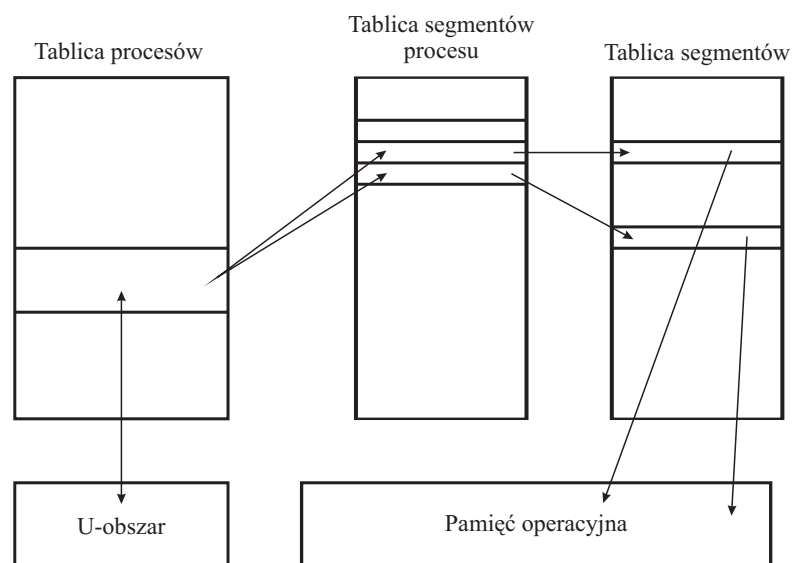
Stany procesu, których dokładny opis przedstawiono w następnym podrozdziale oraz jego charakterystykę, przede wszystkim z punktu wykorzystania zasobów systemowych, przechowywane są w u-obszarze. Przechowywane w nim są:

- Wskaźnik do odpowiedniego elementu tablicy procesów opisującego ten u-obszar.
- Rzeczywisty (*real*) i obowiązujący (*effective*) identyfikator użytkownika, pozwalające na określenie praw dostępu procesu do innych procesów lub plików w systemie.
- Pola liczników rejestrujące czas spędzony przez dany proces i jego procesy potomne w trybie użytkownika i trybie jądra.
- Tablica obsługi sygnałów, zawierająca adresy funkcji ich obsługi. Zmianę funkcji odpowiedzialnych za ich obsługę dokonuje funkcja systemowa *signal*.
- Pole terminala identyfikujące terminal, z którego proces został uruchomiony.
- Ścieżkę dostępu do bieżącego katalogu.
- Pole pozwalające na detekcję błędu wykonania przez proces funkcji systemowej.
- Tablicę deskryptorów plików otwartych przez proces.
- Pola opisujące graniczne wartości niektórych atrybutów procesu, jak np. jego maksymalny rozmiar, maksymalny możliwy do wykorzystania czas procesora, maksymalny rozmiar pliku, jaki może on zapisać, itd.
- Pola umożliwiające modyfikację ustawiania praw dostępu do plików tworzonych przez proces.

Jądro ma dostęp do u-obszaru jedynie aktualnie wykonywanego procesu. Kiedy nastąpi zmiana aktualnie wykonywanego procesu, jądro zmienia organizację swoich struktur dla uzyskania dostępu do u-obszaru tegoż procesu. Znajdujący się w niej wskaźnik do odpowiedniej tablicy procesów pozwala uzyskać pełny dostęp do struktur opisujących ten proces.

Jak wspomniano, rekord opisujący proces w tablicy procesów zawiera wskaźnik na tablicę segmentów procesu. Pozycje w tej tablicy zawierają z kolei wskazania do tablicy segmentów. Pozycje w tablicy segmentów zawierają informacje o tym, co zawiera segment, czy jest prywatny czy dzielony oraz wskazanie do miejsca w pamięci, w którym się znajduje. Dodatkowy stopień pośredniości, prowadzący z tablicy segmentów procesu do tablicy segmentów, umożliwia procesom współdzielenie segmentów, gdyż w praktyce na jedną pozycję w tablicy segmentów mogą wskazywać pola z kilku tablic segmentów procesu. Dotyczy to przede wszystkim segmentu kodu, który może być współdzielony między wieloma procesami w celu optymalizacji wykorzystania pamięci operacyjnej.

Z procesem związane jest pojęcie *kontekstu*. Jest to jego stan opisany wartościami zmiennych globalnych i struktur danych użytkownika, wartości zapisane przez niego w rejestrach procesora, wartości przechowywane w polach rekordu tablicy procesów i odpowiadającego mu u-obszaru oraz wartości odłożone na jego stosy dla pracy w trybie użytkownika i trybie jądra wraz z wartościami ich wskaźników. Mówi się, że podczas wykonywania danego procesu jądro systemu operacyjnego wykonuje się w kontekście tego procesu. Ze zmianą kontekstu mamy do czynienia



Rysunek 4.2: Struktury danych systemu operacyjnego wykorzystywane do obsługi procesów.

wówczas, gdy zmianie ulegnie aktualnie wykonywany proces. Jądro dokonuje wówczas przełączenia kontekstu i zaczyna się wykonywać w kontekście nowego procesu. Zmiana kontekstu jest możliwa jedynie pod pewnymi warunkami. Przełączanie kontekstu to oczywiście wykonywanie funkcji jądra (praca w trybie jądra). Jednak jądro zachowuje niezbędną informację aby móc powrócić do trybu użytkownika i kontynuować wykonywanie w kontekście procesu, którego wykonywanie zostało przerwane. Zaznaczmy, iż przejście między trybem użytkownika a trybem jądra jest zmianą trybu i odbywa się w ramach wykonania tego samego procesu. Zmiana kontekstu to zmiana wykonywanego procesu. Przerwania obsługiwane są przez jądro w ramach aktualnie wykonywanego procesu nawet jeśli to nie on spowodował wygenerowanie przerwania oraz niezależnie od tego, czy wykonywał się on w trybie jądra czy w trybie użytkownika. Jądro systemu zachowuje wówczas niezbędną informację, aby móc powrócić do wykonywania przerwanych procesu. Samo zaś obsługuje przerwanie pracując w trybie jądra (nie uruchamia procesu do obsługi przerwania).

#### 4.1.2 Stany procesu

Życie procesu w systemie operacyjnym Unix można umownie podzielić na stany. Pełna lista możliwych stanów procesu jest następująca:

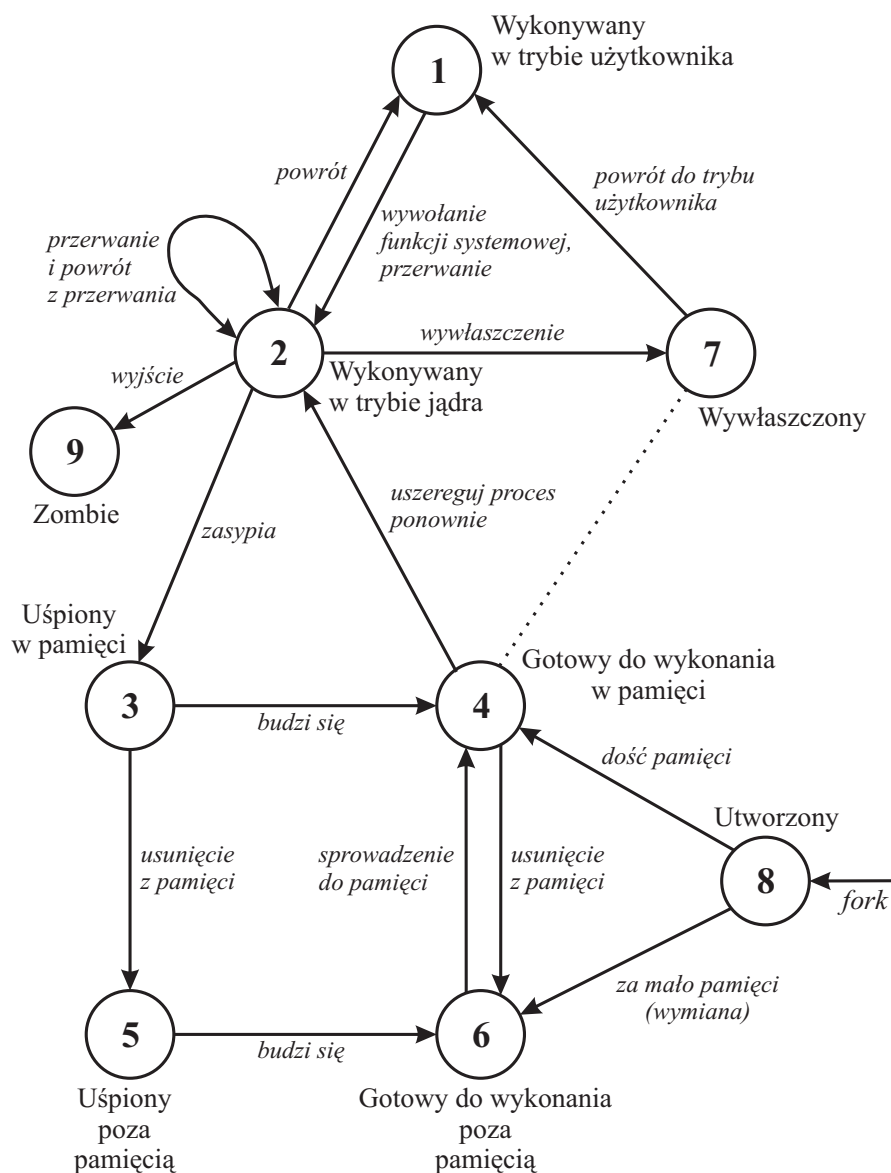
1. Proces wykonuje się w trybie użytkownika.
2. Proces wykonuje się w trybie jądra.
3. Proces śpi i znajduje się w pamięci operacyjnej.
4. Proces nie wykonuje się, ale znajduje się w kolejce zadań gotowych do wykonania czekając, aż jądro przydzieli mu procesor.
5. Śpi, a proces obsługi pamięci wirtualnej przesłał go do pliku wymiany aby w pamięci głównej uzyskać miejsce dla innego procesu.

6. Jest gotów do wykonania lecz znajduje w pliku wymiany. Dlatego proces obsługi pamięci wirtualnej musi przenieść go do pamięci operacyjnej, zanim jądro przydzieli mu procesor do wykonywania jego instrukcji.
7. Proces przechodzi z trybu jądra do trybu użytkownika, lecz jądro wywłaszcza go i przełącza kontekst aby wybrać do wykonania inny proces.
8. Proces jest procesem nowoutworzonym i znajduje się w stanie przejściowym. Zostały zainicjalizowane wszystkie opisujące je struktury, ale proces nie jest jeszcze gotowy do wykonania np. z powodu ładowania zawartości segmentów z pliku. To typowy stan początkowy dla wszystkich procesów za wyjątkiem procesu o numerze 0.
9. Proces wykonał funkcję systemową *exit*, lecz jest w stanie zombie. Segmenty do niego należące oraz u-obszar oraz zostały zwolnione. Pozostał jedynie rekord zawierający kod wyjścia, który zostanie odebrany przez proces macierzysty. Stan ten jest końcowym stanem procesu.

Na rysunku 4.3 przedstawiono pełny diagram przejść międzystanowych procesu. Jest to graf skierowany, w którym wierzchołki przedstawiają stany (numery zgodne z powyższą listą) zaś krawędzie opisane zostały przez zdarzenia powodujące przejście pomiędzy stanami. Stany opisano wykorzystując czcionkę normalną, zaś zdarzenia nieco mniejszą czcionką pochyłą. Oczywiście nie każdy proces w czasie swojego istnienia przechodzi przez wszystkie, opisane stany. Również nie wszystkie stany mogą zostać zaobserwowane, ze względu na krótki czas przebywania w nich procesu. Dodatkowo, przebywanie procesów w stanach numer 5 i 6 wskazują na niedobór pamięci operacyjnej w systemie komputerowym, co negatywnie odbija się na jego wydajności.

Przeanalizujemy możliwe stany procesu. Proces pojawia się w systemie po wykonaniu przez proces macierzysty funkcji systemowej *fork* (stan 8), następnie staje się procesem gotowym do wykonania (stan 4 lub 6). Na początku założymy, że w systemie istnieje wystarczająco dużo wolnej pamięci operacyjnej dla utworzenia jego segmentów. Proces kolejukujący przydzieli utworzonemu procesowi procesor i przejdzie on do stanu „wykonywany w trybie jądra”, w którym zakończy wykonywanie swojej części funkcji *fork*. Wykonanie funkcji systemowej *exec* spowoduje, iż jądro systemu zapisze w segmentach procesu odpowiednie dane. Po zakończeniu wykonywania funkcji systemowej, proces może przejść do stanu „wykonywany w trybie użytkownika”. Po wykorzystaniu przydzielonego procesowi kwantu czasu, jego praca zostanie przerwana i przejdzie on do stanu „wykonywany w trybie jądra”. Jeśli proces kolejukujący zdecyduje o przydzieleniu procesora innemu procesowi, rozważany proces przejdzie do stanu „wywłaszczone”. W przeciwnym przypadku otrzyma on ponownie procesor i przejdzie do stanu „wykonywany w trybie użytkownika”. Stan „wywłaszczone” oraz „gotowy do wykonania w pamięci” są równoważne, co na rysunku zaznaczono punktowaną linią. Ich rozdzielenie ma za zadanie zaznaczenie faktu, iż proces wykonywany w trybie jądra może być wywłaszczone tylko tuż przed powrotem do pracy w trybie użytkownika. Jądro mogłoby zatem w razie braku pamięci operacyjnej usunąć z niej proces będący w stanie wywłaszczone”. Proces kolejukujący przydziela procesor procesowi, który wraca do stanu „wykonywany w trybie użytkownika”.

Podjęcie wykonywania przez proces funkcji systemowej oznacza przejście od stanu „wykonywany w trybie użytkownika” do stanu „wykonywany w trybie jądra”. Dla dalszych rozważań założymy, że wywołana funkcja systemowa żąda wykonania operacji wejścia/wyjścia i proces musi czekać na zakończenie transmisji. Przechodzi on zatem do stanu „uśpiony w pamięci” i znajduje się w nim do momentu, w którym pojawi się przerwanie informujące o zakończeniu operacji wejścia/wyjścia. W wyniku obsługi przerwania proces zostanie obudzony i przejdzie do stanu „gotowy do wykonania w pamięci”. Proces kończący działanie wywołuje funkcję systemową *exit*, przechodząc przez stan „wykonywany w trybie jądra” trafia do stanu „zombie”.



Rysunek 4.3: Graf stanów procesu.

### 4.1.3 Szeregowanie zadań

System operacyjny Unix posiada mechanizmy pozwalające na utrzymywanie wielu procesów w gotowości do wykonania oraz odpowiednie przydzielanie im procesora. Dla potrzeb dalszych rozważań założymy, iż procesor jest pojedynczym zasobem i w danym momencie czasu może wykonywać instrukcje jednego procesu. Najprostsze rozwiązanie polega na umieszczeniu wszystkich procesów w kolejce, w której będą czekały na przydzielenie procesora. Kolejność procesów, wyznaczana w ściśle określonych momentach czasu, będzie zależała od wartości wybranych atrybu-

tów procesu. Mogą nimi być przykładowo sumaryczny czas wykonywania się procesu w systemie, intensywność wykonywania operacji wejścia/wyjścia, wykorzystanie pamięci operacyjnej i inne. Sposób wyznaczania wartości charakteryzującej dany proces, zwanej *priorytetem*, stanowi część algorytmu szeregowania, charakterystycznego dla danej implementacji systemu.

W niniejszym podrozdziale, w pierwszej kolejności, przedstawione zostaną mechanizmy kolejowania zadań w systemach BSD. Są one bardzo intuicyjne i w prosty sposób pozwolą zrozumieć istotę działania tego podsystemu. W dalszej kolejności omówione zostaną modyfikacje wprowadzone w SVR4. Stanowią one próbę uczynienia systemu kolejowania procesów dostępnego do modyfikacji bez konieczności posiadania dostępu do kodu źródłowego jądra systemu operacyjnego. Zakres modyfikacji przy takim podejściu jest oczywiście ograniczony.

### Szeregowanie zadań w systemach BSD

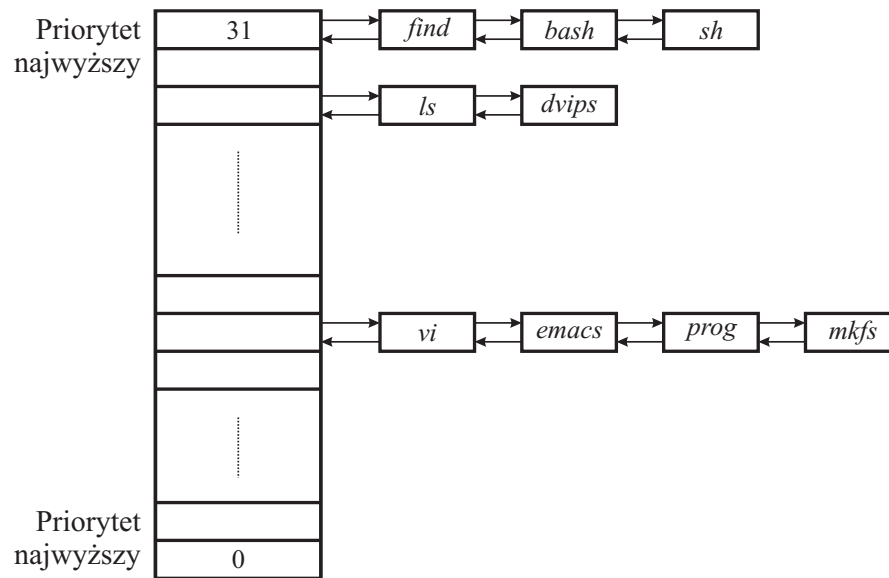
Na rysunku 4.4 przedstawiono organizację kolejki w systemach BSD. Została ona zaimplementowana w postaci 32-dwu elementowej tablicy, której każdy element będący rekordem, zawiera m.in. wskaźnik do listy procesów posiadających wartość liczbowa priorytetu mieszczącą się w tym samym zakresie. Jądro systemu poszukuje procesu do wykonania, przeglądając listy od góry do dołu. Stąd proces o wyższym priorytecie ma pierwszeństwo przed procesem o niższym priorytecie. Należy zwrócić uwagę, iż wartość liczbowa priorytetu procesu posiadającego wyższy priorytet jest mniejsza od wartości liczbowej priorytetu procesu o mniejszym priorytecie. Jeśli w danej kolejce znajduje się wiele procesów, to jądro wykonuje każdy z nich przez określony czas zwany kwantem. W klasycznej implementacji systemów BSD kwant czasu jest stały i wynosi 0.1s. Taki mechanizm szeregowania zadań jest zazwyczaj nazywany *round-robin*, gdyż wszystkie procesy o takiej samej wartości priorytetu traktowane są tak samo. Dlatego jądro nie przydzieli procesora dla procesu z kolejki o niższym priorytecie dopóty, dopóki kolejki zadań o wyższym priorytecie nie będą puste. W praktyce oznacza to, że procesy o wyższym priorytecie będą działały do momentu, w którym przestaną potrzebować procesora, co najczęściej ma miejsce przy oczekiwaniu na zakończenie operacji wejścia/wyjścia (proces w stanie „uśpiony”). Drugą możliwością jest zmniejszenie priorytetu spowodowane długotrwałym wykonywaniem.

W przedstawionej na rysunku 4.4 implementacji, procedura szeregowania wykorzystuje 32 kolejki przy 256 wartościach priorytetu. Każda kolejka posiada taką samą liczbę wartości priorytetu. Stąd niewielkie zmiany wartości priorytetu (do 7) spowodują, iż proces nadal będzie znajdował się w tej samej kolejce. Dlatego niektóre systemy wprowadziły większą liczbę kolejek. Przykładowo od wersji 4 systemu AIX jest ich 128.

Algorytmy kolejowania, które pojawiły się w każdej implementacji systemu Unix, zaprojektowano w taki sposób, aby były „sprawiedliwe”, a nie z myślą o priorytetyzowaniu procesów wybranej grupy użytkowników. Dlatego żaden proces, niezależnie od jego priorytetu, starający się o dostęp do procesora nie zostanie zagłuszony.

### Szeregowanie zadań w Systemie V

Jak wcześniej wspomniano, funkcje szeregowania procesów w systemie Unix były „zaszyte” w jądrze systemu operacyjnego. W praktyce ich modyfikacja polegała na ingerencji w kod źródłowy jądra oraz jego kompilację. Zazwyczaj czynności te nie były konieczne, gdyż opracowane na podstawie wieloletnich doświadczeń, algorytmy kolejowania dobrze sprawdzały się w przeciętnych systemach wielodostępnych. Jednak w zastosowaniach dedykowanych często okazywało się, iż udostępniona do modyfikacji elastyczność algorytmów kolejowania jest niezadowalająca. Stąd w systemie SVR4 zaimplementowano modularny mechanizm szeregowania zadań, który pozwala na wybór procedury szeregowania indywidualnie dla każdego procesu.



Rysunek 4.4: Organizacja kolejki procesów gotowych do wykonania w systemie BSD.

Klasyczna implementacja systemu SVR4 udostępnia trzy procedury kolejkowania zadań. Są to: TS (*Time Sharing*), RT (*Real Time*) oraz SYS (*System*). Wartości atrybutów procedur kolejkowania TS oraz RT można zmieniać wykorzystując w tym celu program *pricntl*. Istnieje możliwość usunięcia istniejących procedur kolejkowania oraz oczywiście dodania własnych. Listę procedur kolejkowania, które są aktualnie dostępne uzyskamy poleceniem *dispadmin* uruchomionym z opcją **-l**. W systemie jest ona następująca:

```

1  bash-2.05$ pricntl -l
2  CONFIGURED CLASSES
3  =====
4
5  SYS (System Class)
6
7  TS (Time Sharing)
8      Configured TS User Priority Range: -60 through 60
9
10  FX (Fixed priority)
11      Configured FX User Priority Range: 0 through 60
12
13  RT (Real Time)
14      Maximum Configured RT Priority: 59
15
16  IA (Interactive)
17      Configured IA User Priority Range: -60 through 60

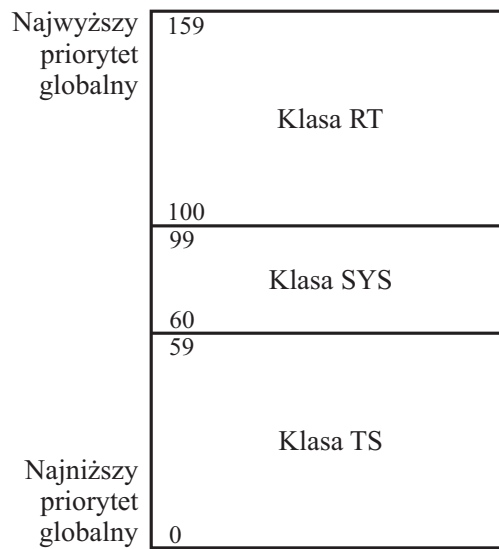
```

Procedurę kolejkowania TS stosuje się w przeciętnych systemach pracujących w trybie z podziałem czasu. Procedura kolejkowania RT jest przeznaczona dla procesów czasu rzeczywistego.

W zakres procedury SYS wchodzi przydzielanie priorytetów systemowych. W rozpatrywanym systemie (SunOS 5.9) dostępne są również kolejki FX (*Fixed Priority*) oraz IA (*Interactive*).

### Przeliczanie priorytetów globalnych w systemie SVR4

Względne zależności priorytetów klas szeregowania w SVR4 przedstawiono na rysunku 4.5. Klasa SYS obejmuje faktycznie najniższy zakres globalnych priorytetów, zaś klasy RT oraz TS korzystają z dwóch większych segmentów. Stąd w praktyce procesy czasu rzeczywistego (RT) mają priorytety wyższe niż dowolny proces klasy TS, jeśli nawet wykonuje się w trybie jądra.



Rysunek 4.5: Wartości priorytetów klas szeregowania w systemie SVR4.

W implementacji zaproponowanej w systemie SVR4 istnieje jeden globalny, obejmujący wszystkie klasy priorytetów, zbiór wartości priorytetów, wykorzystywanych przez jądro do kolejki zadań. Klasa szeregowania procesu jest dziedziczona przez proces potomny przy wywołaniu funkcji systemowej *fork*. Wszystkie procesy uruchamiane w systemie używają domyślnej klasy TS, gdyż program */etc/init*, będący ich przodkiem używa właśnie tej klasy kolejki.

### Klasy TS

Procedura kolejki TS została zaprojektowana tak, aby mogła działać podobnie to tradycyjnego algorytmu szeregowania stosowanego dotychczas w systemach Unix. Istnieje jednak wiele znaczących różnic. Najistotniejsza polega na istnieniu zmiennego kwantu czasu, który jest tym krótszy, im większy jest priorytet procesu. Kolejną różnicą jest taka, iż większe liczbowo wartości priorytetu oznaczają wyższy priorytet procesu. Jednak interfejs użytkownika używa tradycyjnego systemu, w którym większa wartość liczbowo oznacza niższy priorytet. Poniżej przedstawiono wartości atrybutów procedury TS w systemie :

```

1  bash-2.05$ /usr/sbin/dispatcher -c TS -g
2  # Time Sharing Dispatcher Configuration
3  RES=1000

```

	#	ts_quantum	ts_tqexp	ts_slpret	ts_maxwait	ts_lwait	PRIORITY	LEVEL
4								
5								
6		200	0	50	0	50	#	0
7		200	0	50	0	50	#	1
8		200	0	50	0	50	#	2
9		200	0	50	0	50	#	3
10		200	0	50	0	50	#	4
11		200	0	50	0	50	#	5
12		200	0	50	0	50	#	6
13		200	0	50	0	50	#	7
14		200	0	50	0	50	#	8
15		200	0	50	0	50	#	9
16		160	0	51	0	51	#	10
17		160	1	51	0	51	#	11
18		160	2	51	0	51	#	12
19		160	3	51	0	51	#	13
20		160	4	51	0	51	#	14
21		160	5	51	0	51	#	15
22		160	6	51	0	51	#	16
23		160	7	51	0	51	#	17
24		160	8	51	0	51	#	18
25		160	9	51	0	51	#	19
26		120	10	52	0	52	#	20
27		120	11	52	0	52	#	21
28		120	12	52	0	52	#	22
29		120	13	52	0	52	#	23
30		120	14	52	0	52	#	24
31		120	15	52	0	52	#	25
32		120	16	52	0	52	#	26
33		120	17	52	0	52	#	27
34		120	18	52	0	52	#	28
35		120	19	52	0	52	#	29
36		80	20	53	0	53	#	30
37		80	21	53	0	53	#	31
38		80	22	53	0	53	#	32
39		80	23	53	0	53	#	33
40		80	24	53	0	53	#	34
41		80	25	54	0	54	#	35
42		80	26	54	0	54	#	36
43		80	27	54	0	54	#	37
44		80	28	54	0	54	#	38
45		80	29	54	0	54	#	39
46		40	30	55	0	55	#	40
47		40	31	55	0	55	#	41
48		40	32	55	0	55	#	42
49		40	33	55	0	55	#	43
50		40	34	55	0	55	#	44
51		40	35	56	0	56	#	45
52		40	36	57	0	57	#	46



53	40	37	58	0	58	#	47
54	40	38	58	0	58	#	48
55	40	39	58	0	59	#	49
56	40	40	58	0	59	#	50
57	40	41	58	0	59	#	51
58	40	42	58	0	59	#	52
59	40	43	58	0	59	#	53
60	40	44	58	0	59	#	54
61	40	45	58	0	59	#	55
62	40	46	58	0	59	#	56
63	40	47	58	0	59	#	57
64	40	48	58	0	59	#	58
65	20	49	59	32000	59	#	59

Zawartość linii 3 powyższego listingu informuje nas, iż jednostka czasu to 0.001s czyli milisekunda. Stąd przykładowa wartość 160 oznacza w praktyce 0.16s. Mechanizm kolejkowania używa własnego zegara, w którym jednostka to 0.01s. Stąd rzeczywista rozdzielczość wynosi 10 milisekund. Idąc dalej, wartości priorytetu podane są w kolumnie oznaczonej nagłówkiem *PRIORITY LEVEL*. Przyjmują one wartości z przedziału od 0 (najniższy) do 59 (najwyższy). Długość kwantu czasu zawiera kolumna opisana jako *ts\_quantum*. Dla najwyższej wartości priorytetu przyjmuje on wartość 0.04s zaś dla najniższej 0.2s. Mechanizm migracji priorytetu jest bardzo prosty. Kolumna *ts\_tqexp* zawiera wartość priorytetu, jaką otrzyma proces po wykorzystaniu procesora przez przydzielony mu kwant czasu. Załóżmy, że uruchamiamy proces z domyślną wartością priorytetu 30. Otrzyma on procesor na 0.08s, a po jego wykorzystaniu nowa wartość priorytetu zostanie ustalona na 20 (linia 36). Jeśli proces kolejkowania wybierze rozpatrywany proces do wykonania, to procesor zostanie mu przydzielony na 0.12s, a po tym czasie wartość priorytetu wyniesie 10 (linia 26). Stąd kolejny kwant czasu wyniesie 0.16s, a nowa wartość priorytetu 0. Od tego momentu proces będzie otrzymywał procesor na 0.2s, a nowa wartość priorytetu będzie wynosić 0. Oczywiście na zmianę wartości priorytetu mają wpływ inne czynniki, które omówimy w dalszej części. W przytoczonym przykładzie zostały one pominięte dla uproszczenia rozważań.

Założenie, iż procesy o wyższym priorytecie mają mniejszy kwant czasu od procesów posiadających niższy priorytet wydaje się być sprzeczne z intuicją. Praktycznie, dla procesów o dużym priorytecie nie ma to większego znaczenia, gdyż otrzymują one procesor bardzo często, a zatem sumaryczny czas, w którym procesor będzie wykonywał ich instrukcje będzie duży. Inaczej sytuacja wygląda w przypadku procesów o niskim priorytecie. Dla wyjaśnienia rozważmy sytuację, w której kilka takich procesów pracuje w naszym systemie. Ponieważ niezależnie od rozmiaru kwantu czasu procesy o niskim priorytecie wykonują się rzadko, więc prawdopodobieństwo, że tablice systemowe oraz wszelkie pamięci podręczne nie zawierają danych przyspieszających ich działanie jest duże. Stąd przy każdym przydzieleniu im procesora upływa wiele czasu, zanim w pamięciach podręcznych pojawią się dane konieczne do działania procesu. Stąd przydzielanie procesom o niskim priorytecie małych kwantów czasu byłoby nieefektywne, gdyż praktycznie byłyby one zużywane jedynie na uzupełnianie zawartości pamięci podręcznej. Natomiast procesy o wysokim priorytecie wykonywane są często i pamięci podręczne z dużym prawdopodobieństwem zawierają stosowne dane. Kwant czasu jest niemal w pełni wykorzystywany do wykonywania instrukcji procesu.

Inną zaletą wynikającą z zastosowania dłuższych kwantów czasu dla procesów o niższym priorytecie jest ograniczenie liczby wykonywanych operacji przełączania kontekstu, które są dość kosztowne. Przykładowo wykonanie dwóch współbieżnych procesów z kwantem czasu 0.04s daje 25 przełączeń na sekundę. Kwant czasu 0.2s pozwala ograniczyć tę liczbę do 5.

W przypadku procesów interaktywnych o dużym priorytecie, korzyści płynące z zastosowania dłuższego kwantu czasu nie są aż tak widoczne. Wydłużenie kwantu czasu powoduje wydłużenie czasu reakcji, gdyż system rzadziej przełącza się między procesami. Stąd iluzja równoczesnego wykonywania procesów staje się mniej wyrazista.

```

1 bash-2.05$ priocntl -d $$
2 TIME SHARING PROCESSES:
3     PID      TSUPRILIM    TSUPRI
4     20826         0         0

```

## 4.2 Plik */etc/inittab*

## 4.3 Crontab

## 4.4 Obsługa plików dziennika

Mechanizm rejestrowania zdarzeń zachodzących w systemach uniksowych jest ich bardzo mocną stroną. Istotną jego cechą są szerokie możliwości konfiguracyjne, umożliwiające rejestrowanie właściwie dowolnych zdarzeń z wybranym poziomem szczegółowości. W konfiguracji domyślnej rejestrowana jest aktywność najczęściej używanych zasobów systemu. W praktyce, dla każdego systemu indywidualnie, administrator wybiera konfigurację, która najlepiej odpowiada jego potrzebom.

### 4.4.1 Rejestrowanie zdarzeń w dystrybucji Fedora

### 4.4.2 Rejestrowanie zdarzeń w dystrybucji FreeBSD

W tej rodzinie systemów do prowadzenia dzienników wykorzystuje się proces demona *syslogd*. Proces ten obsługuje komunikaty według tzw. kanału (ang. *facility*) i poziomu (ang. *level*). Wartości tych parametrów są dołączane do każdego komunikatu wysyłanego do procesu *syslogd*. Zaczniemy od opisu tych parametrów.

#### Kanały

Kanał to identyfikator rodzaju komunikatu, który został przesłany do procesu *syslogd*. W praktyce jest on łańcuchem znaków stanowiącym jego opis. Większość aplikacji, które wymagają oddzielnych dzienników posiada własny kanał. Zatem do wybranego pliku dziennika będą zapisywane informacje pochodzące z jednego kanału. Proces *syslogd* ma do dyspozycji kilka uniwersalnych kanałów, które mogą być wykorzystywane przez dowolne aplikacje. Aplikacje do komunikatów dołączają swoje nazwy. Stąd jeśli nawet kilka aplikacji wykorzystuje ten sam kanał, to istnieje możliwość odnalezienia właściwych wykorzystując tę właśnie informację. Standardowe kanały wraz z ich krótkim opisem przedstawiono w tabeli 4.4.2.

#### Poziomy komunikatów

Poziom komunikatu decyduje o jego względnej ważności. Procesy przesyłają wszystkie komunikaty do demona *syslogd*. Ten filtruje je zgodnie ze swoją konfiguracją, zapisując wskazane do plików dziennika, zaś pozostałe odrzuca. Oczywiście *istotność* jest pojęciem względnym. Stąd

Kanał	Opis
<i>auth</i>	Kanał jest wykorzystywany do przekazywania publicznie dostępnych informacji dotyczących autoryzacji użytkowników czyli połączeń do systemu oraz zmian tożsamości.
<i>authpriv</i>	Kanał służy do przekazywania poufnych informacji dotyczących autoryzacji użytkowników w systemie. Informacja ta jest dostępna dla użytkownika <i>root</i> .
<i>console</i>	Kanał jest przeznaczony do rejestracji zdarzeń, o których komunikaty pojawiają się zazwyczaj na konsoli systemowej.
<i>cron</i>	Kanał dla komunikatów programu <i>cron</i> .
<i>daemon</i>	Jest to kanał wspólny dla wszystkich demonów systemowych, które nie mają własnych programów obsługi.
<i>ftp</i>	Z kanału korzystają demony protokołu FTP (ang. <i>File Transfer Protocol</i> ).
<i>kern</i>	Kanał rejestruje komunikaty jądra systemu operacyjnego.
<i>lpr</i>	Kanał jest przeznaczony do rejestrowania zdarzeń pochodzących z systemu drukowania.
<i>mail</i>	Kanał rejestruje zdarzenia pochodzące z systemu pocztowego.
<i>mark</i>	Kanał ten nie służy do rejestrowania komunikatów pochodzących z systemu, ale powoduje, że co 20 minut do odpowiedniego dziennika jest wstawiany wpis (znacznik czasowy).
<i>news</i>	Kanał gromadzi komunikaty demonów obsługujących grupy dyskusyjne.
<i>ntp</i>	Kanał wykorzystywany do gromadzenia informacji o pracy sieciowego protokołu czasu (ang. <i>Network Time Protocol</i> ).
<i>security</i>	Służy do przekazywania informacji o różnym poziomie szczególności związanych z bezpieczeństwem systemu. Wykorzystywany np. przez filtry <i>ipf</i> i <i>ipfw</i> .
<i>syslog</i>	Kanał wykorzystywany przez usługę do rejestracji zdarzeń o sobie samej.
<i>user</i>	Uniwersalny kanał rejestrujący zdarzenia pochodzące od programów użytkownika.
<i>uucp</i>	Kanał przeznaczony do rejestrowania komunikatów dotyczących protokołu <i>UUCP</i> . W chwili obecnej praktycznie nie wykorzystywany.
<i>local10–local17</i>	Kanały zarezerwowane na użytek administratora, możliwe do wykorzystania przez programy posiadające opcje wyboru kanału rejestrującego.

Tablica 4.1: Standardowe kanały zdarzeń w dystrybucji FreeBSD.

możliwość własnej konfiguracji. W systemie FreeBSD mamy dostępnych osiem poziomów ważności komunikatów. Ich opis zawarto w tabeli 4.4.2, w kolejności od najbardziej do najmniej ważnego.

Poziom	Opis
<i>emerg</i>	
<i>alert</i>	
<i>crit</i>	
<i>err</i>	
<i>warning</i>	
<i>notice</i>	
<i>info</i>	
<i>debug</i>	
<i>none</i>	

Tablica 4.2: Poziomy ważności komunikatów w dystrybucji FreeBSD.

## Rozdział 5

# Interpretery poleceń

Interpreter poleceń pełni rolę pośrednika między użytkownikiem, a jądrem systemu operacyjnego. Stąd znajomość jego funkcjonalności przekłada się bezpośrednio na efektywność pracy użytkownika w systemie. Szczególnie istotne są zagadnienia związane z konfigurowaniem sposobu pracy interpretera oraz wykorzystanie jego języka programowania. Interpreter poleceń może teoretycznie stworzyć każdy, kto zna protokół komunikacji z jądrem, a jest on dobrze udokumentowany, oraz język programowania. Dlatego też powstało wiele interpreterów, poleceń lecz tylko kilka zyskało popularność. W niniejszym rozdziale zostaną omówione te najczęściej występujące w systemach uniksowych.

### 5.1 Systematyka

Podział interpreterów poleceń można przeprowadzić ze względu na dwa kryteria. Pierwszym jest funkcja, jaką pełni dany interpreter. Można tu wyróżnić interpreter logujący, dzięki któremu jesteśmy podłączeni do systemu, interpreter interaktywny uruchomiony z linii poleceń oraz interpreter uruchomiony dla zinterpretowania skryptu. Drugie kryterium dotyczy języka programowania, a właściwie jego podobieństwa do języka C. Wyróżniamy zasadniczo interpretry wyposażone w język, którego składnia jest podobna do składni języka C i pozostałe. W systemach uniksowych najpopularniejsze są cztery odmiany interpreterów, a to: interpreter Bourne’a, interpreter Korn’a, interpreter C (popularnie zwany *cs**h*) oraz stanowiący jego rozwinięcie interpreter *tcsh*. Domyślnym interpreterem systemów linuxowych jest interpreter *bash*, który staje się coraz częściej wykorzystywany również w systemach uniksowych.

Interpreter Bourne’a jest najczęściej spotykanym interpreterem, występującym w podstawowej instalacji właściwie każdego systemu uniksowego i linuxowego. Pierwsza wersja pojawiła się około roku 1975. Jego atutem jest prostota, bezpośrednio przekładająca się na efektywność działania. Nie posiada on jednak funkcji interaktywnych oraz dopełniania wiersza poleceń. Mechanizm przekierowania danych z jednego polecenia do następnego (*pipe*) jest bardzo elastyczny i umożliwia przekierowanie standardowego wejścia i wyjścia całej konstrukcji sterującej. Do komunikacji między procesami udostępnia potoki nazwane. W programach istnieje możliwość wykorzystywania zmiennych lokalnych i globalnych, które mogą być przekazywane do środowiska procesu potomnego (eksportowane). Obsługa wyjątków w interpreterze Bourne’a jest realizowana poleceniem *trap*. Do obliczania wartości wyrażeń arytmetycznych służy zewnętrzne polecenie *expr*, zaś wartości wyrażeń logicznych wyznacza się poleceniem *test* lub *expr*.

Interpreter Korn’a posiada pełną funkcjonalność interpretera Bourne’a, uzupełnioną o kilka rozszerzeń charakterystycznych dla interpretera *cs**h*. Pojawiły się w nim mechanizmy edycji wierszy

sza polecenia zrealizowane przez najpopularniejsze edytory. Udoskonalono mechanizmy przechowywania wydanych poleceń, co umożliwiło bezpośrednie ich przywoływanie z listy. Wartości wyrażeń logicznych obliczane są przez interpreter poleceń. Poszerzono mechanizmy sterowania zadaniami planu pierwszego i tła. W interpreterze Bourne'a zadanie uruchomione w tle lub na planie pierwszym pozostawało tam do końca wykonania, bez możliwości przenoszenia. Interpreter Korna umożliwia przenoszenie zadań.

Interpreter C opracowano na Uniwersytecie Kalifornijskim w Berkeley. Dla podniesienia wydajności wyposażony został w długą listę poleceń wbudowanych. Również wartości wyrażeń logicznych obliczane są przez interpreter. Interpreter pamięta również listę wydanych poleceń, umożliwia ich przywoływanie, modyfikację i wykonanie. Zaimplementowano w nim także mechanizm aliasów umożliwiający nadawanie krótkich nazw zawiłym poleceniom, co ułatwia i przyspiesza pracę z systemem. Interpreter posiada mechanizmy zarządzania procesami w tym uruchamianie ich na planie pierwszym i w tle oraz możliwość przenoszenia ich między planem pierwszym i tłem. Interpreter posiada dwa rodzaje zmiennych, zwykle (lokalne) oraz środowiskowe (globalne).

Interpreter *tcs*h, jak wspomniano, stanowi rozwinięcie interpretera *csh*, a litera *t* w nazwie pochodzi od słowa TENEX, nazwy systemu operacyjnego, który zainspirował autora do unowocześnienia interpretera *csh* poprzez dodanie mechanizmu uzupełniania nazw poleceń oraz plików w linii poleceń. Powłoka ta jest domyślną w systemach BSD.

Nazwa interpretera *bash* jest akronimem od Bourne–Again Shell (angielska gra słów: fonetycznie brzmi tak samo, jak born again shell, czyli odrodzona powłoka). Jak łatwo można się domyśleć, wywodzi się on z interpretera Bourne'a, ale zawiera niemal wszystkie funkcjonalności interpreterów Korna oraz *tcs*h. Do podstawowych należy zaliczyć pełną edycję linii poleceń oraz obsługę historii poleceń. Zaimplementowano w nim także specyficzne rozszerzenia, jak np. zmienne \$RANDOM i \$PPID oraz POSIX-owe podstawianie polecenia (składnia: \$(...) ). Oprócz aliasów istnieje możliwość definiowania funkcji, a zatem przekazywania argumentów, na których będą one działać.

Wśród mniej znanych interpreterów, uwagę zwraca interpreter *zsh* (Z shell). Umożliwia tradycyjną pracę interaktywną. Został także wyposażony w nowoczesny język skryptowy. Jest najbardziej zbliżony do interpretera Korna, zawiera jednak wiele modyfikacji. Do najistotniejszych należą: edycja wiersza poleceń, wbudowana korekta pisowni, programowalne dopełnianie poleceń, możliwość definiowania funkcji oraz obsługę historii poleceń. Lista detali liczy ok. 100 pozycji. Podczas pierwszego uruchomienia interpreter uruchamia program, umożliwiający stworzenie konfiguracji oraz zapisanie jej w odpowiednich plikach. W chwili obecnej projekt jest rozwijany siłami członków listy dyskusyjnej *zsh-workers*.

## 5.2 Działanie

Interpreter poleceń może zasadniczo pracować w dwóch trybach. Najczęściej wykorzystywanym jest tryb interaktywny, służący do uruchamiania procesów specyfikowanych w wierszu polecenia. Interpreter poleceń służy również do wykonywania, a dokładniej interpretowania, programów napisanych w jego języku.

Na początek zajmijmy się pracą interaktywną. Interpreter pracuje cyklicznie, a każdy cykl składa się z następujących kroków:

- Interpreter oczekuje na wprowadzenia polecenia. Polecenie może posiadać opcje i argumenty. Opcje służą do wskazywania sposobu pracy polecenia. Jeśli ich nie użyjemy, polecenie pracuje w sposób domyślny. Opcje poprzedza znak - , który pozwala odróżnić je od argumentów. Argumenty wskazują na obiekty, na jakich ma działać polecenie. Pomińcie

argumentów powoduje, że polecenie pracuje na obiektach domyślnych (np. na zawartości bieżącego katalogu).

Założmy, że użytkownik wprowadził polecenie do wykonania i nacisnął *Enter*. Interpreter sprawdza, czy we wprowadzonym tekście występują *znaki specjalne* i jeśli tak, to zastępuje je zgodnie z ich znaczeniem. Następnie sprawdza, czy powstały w ten sposób wiersz polecenia jest poprawny składniowo. Jeśli tak, to przechodzi do kolejnego kroku. Jeśli nie, to wypisuje komunikat o błędzie i oczekuje na wprowadzenie kolejnego polecenia.

- W drugim kroku interpreter poleceń szuka polecenia, które ma wykonać. Rozpoczyna od listy *poleceń wbudowanych*, gdyż stanowią one kod interpretera i ich wykonanie nie wymaga tworzenia procesu, a zatem jest szybkie i nieobciążające system. Jeśli polecenie zostanie znalezione, to interpreter przystępuje do wykonania właściwego fragmentu kodu. Jeśli nie, przeszukana zostaje lista aliasów. I podobnie, jak w przypadku poleceń wbudowanych, odnalezienie polecenia skutkuje podjęciem próby jego wykonania zgodnie z przepisem umieszczonym w aliasie. Jeśli natomiast polecenie nie zostało znalezione na liście aliasów, to być może jest ono poleceniem zewnętrznym zapisanym w pliku w systemie plików w postaci skryptu lub programu wykonywalnego. Pozostaje kwestia odnalezienia go. Ścieżki dostępu do katalogów, które interpreter poleceń będzie przeglądał w tym celu, oddzielone znakiem `:`, stanowią wartość zmiennej środowiskowej *PATH*. Interpreter poleceń przegląda je w kolejności od lewej do prawej. Znalezienie polecenia skutkuje podjęciem próby jego wykonania i zakończeniem przeglądania katalogów. Jeśli w katalogach określonych wartością zmiennej *PATH* nie zostanie znalezione żądane polecenie, to na ekranie pojawia się stosowny komunikat, interpreter przechodzi do oczekiwania na wprowadzenie kolejnego polecenia.
- Polecenie jest wykonywane. Jeśli uruchomienie polecenia nastąpiło na planie pierwszym, to interpreter poleceń czeka na jego zakończenie i przekazuje skierowane przez nie na standardowe wyjścia znaki. Po zakończeniu wykonywania polecenia przechodzi do oczekiwania na wprowadzenie kolejnego. W przypadku uruchomienia polecenia w tle, interpreter od razu przechodzi w stan oczekiwania, a znaki wypisywane przez pracujące polecenie pojawiają się na standardowych wyjściach w sposób asynchroniczny.

Interpreter poleceń zakłada, że w wierszu polecenia wszystko to, co nie jest poleceniem lub jego opcją, jest plikiem zwykłym, katalogiem lub plikiem specjalnym. Dla łatwiejszego posługiwania się ich nazwami interpretery poleceń rozróżniają tzw. *metaznaki* (*metacharacters*) lub dokładniej *metaznakami generowania nazw plików* (*filename generation metacharacters*). Znaki te służą do zastępowania zwykłych znaków w nazwach plików i katalogów, co pozwala skracać postać linii komend oraz wymusza stosowanie odpowiednich konwencji nazewniczych. Do podstawowych metaznaków należą:

- \* – dopasuj dowolny ciąg znaków z ciągiem pustym włącznie. Przykładowo, wypisanie nazw plików z katalogu `/usr/include` zaczynających się znakami *std*:

```
1 [12:32:06 jan@messy ~]$ ls /usr/include/std*
```

- ? – dopasuj pojedynczy znak alfanumeryczny (nie może to być znak pusty). Przykładowo, wypisanie nazw plików znajdujących się w katalogu `/usr/include`, zaczynających się ciągiem znaków *st* i kończących *.h* o długości 8-miu znaków wymaga użycia polecenia:

```
1 [12:38:31 jan@messy ~]$ ls /usr/include/st?????h
```

[ ... ] – dopasuj pojedynczy znak spośród wymienionych w nawiasach. Stąd wypisanie nazw plików znajdujących się w katalogu `/usr/include`, które zaczynają na litery od *a* do *g* oraz *j* lub *k* umożliwia następujące polecenie:

```
1 13:46 [bory@thorin ~]$ ls -d /usr/include/[a-gjk]*
```

{ ... } – jest to tzw. rozwijanie nawiasów i umożliwia tworzenie nazw plików z wykorzystaniem napisów zawartych w nawiasach na zasadzie iloczynu kartezjańskiego. Przykładowo, wypisanie nazw plików znajdujących się w katalogu `/usr/include`, których nazwy zaczynają się ciągiem znaków *st* lub *co* wymaga użycia polecenia:

```
1 13:48 [bory@thorin ~]$ ls -d /usr/include/{st,co}*
```

Do generowania nazw katalogów interpretery wykorzystują również znak `~`. Jest on interpretowany w/g następujących sposobów:

`~` – zastęp znak bezwzględną ścieżką dostępu do katalogu domowego bieżącego użytkownika.

`~nazwa_logowania` – napis jest zastępowany bezwzględną ścieżką dostępu do katalogu domowego użytkownika, którego nazwa pojawiła się po znaku `.`

`~+` – znaki zastępowane są bezwzględną ścieżką dostępu do bieżącego katalogu roboczego (dostępne w interpreterze *bash*).

`~-` – znaki zastępowane są bezwzględną ścieżką dostępu do poprzedniego katalogu roboczego (dostępne w interpreterze *bash*).

Oprócz znaków generowania nazw plików i katalogów w linii poleceń mogą pojawić się inne znaki specjalne, których opis działania przedstawiono poniżej:

`;` – znak jest separatorem poleceń. W linii poleceń może pojawić się kilka poleceń oddzielonych znakiem `;`. Są one wykonywane od strony lewej do prawej. Wykonanie polecenia następnego nie zależy od tego, czy wykonanie polecenia poprzedniego zakończyło się błędem, czy nie.

`&&` – znak służy do warunkowego wykonywania poleceń. Jeśli w wierszu polecenia pojawi się kilka poleceń „połączonych” tym znakiem, to wykonywane są one od strony lewej do prawej. Jeśli jednak wykonanie któregośkolwiek zakończy się niepowodzeniem, to następne nie są już wykonywane. Przykładowo polecenie:

```
1 gcc -o prog prog.c && ./prog
```

uruchamia program *prog* jeśli jego kompilacja zakończyła się powodzeniem. W przeciwnym wypadku nie jest on uruchamiany.

`||` – ten znak również umożliwia warunkowe wykonanie poleceń. Jeśli w wierszu poleceń pojawi się kilka poleceń „połączonych” znakiem `||`, to wykonywane są one od strony lewej do prawej. Zakończenie wykonywania pierwszego z nich z sukcesem spowoduje, że następne nie będą już wykonywane.



- & – pojawienie się znaku na końcu polecenia oznacza wykonanie go w tle. Jest również wykorzystywany przy przekierowaniu wyjścia diagnostycznego, do odróżniania nazwy pliku od jego numeru.
- | – znak umożliwia skierowanie standardowego wyjścia polecenia, które pojawiło się po jego stronie lewej na standardowe wejście polecenia z jego strony prawej.
- > – przekieruj standardowe wyjście polecenia do pliku, którego nazwa pojawiła się po stronie prawej znaku. Przekierowanie działa w trybie nadpisywania (*overwrite*): jeśli plik o podanej nazwie nie istnieje to jest tworzony. Jeśli istnieje, to jego zawartość jest usuwana, a w jej miejsce pojawia się to, co polecenie skierowało do swojego standardowego wyjścia.
- » – znak umożliwia przekierowanie standardowego wyjścia polecenia do pliku, które nazwa pojawiła się po ich prawej stronie. Przekierowanie działa w trybie dopisywania (*append*): jeśli plik o podanej nazwie nie istnieje, to jest tworzony. Jeśli istnieje, to znaki skierowane przez polecenia do jego standardowego wyjścia są dopisywane na koniec pliku bez usuwania jego dotychczasowej zawartości.
- < – znak umożliwia przekierowanie standardowego wejścia. Praktycznie, polecenie zamiast czytać ze standardowego wejścia (klawiatury) będzie czytało z pliku, którego nazwa pojawiła się po stronie prawej znaku. Przekierowanie to jest wykorzystywane np. podczas testowania programu, gdyż umożliwia jego uruchamianie z tym samym zestawem danych wejściowych, zapisanych w pliku.
- « – znak wykorzystywany jest do oznaczania końca danych czytanych przez aplikację, jeśli ta jest uruchamiana w sposób uniemożliwiający dostęp do standardowego wejścia (np. z wykorzystaniem polecenia *at* lub tablicy demona zegarowego *crontab*). Po stronie prawej znaku pojawia się napis będący oznaczeniem końca danych. Przykładowo, uruchomienie programu czytającego ze standardowego wejścia dwie liczby rzeczywiste może zostać zrealizowane przez skrypt o następującej postaci:

```

1 ./program << EOF
2 1.123
3 0.988
4 EOF

```

- () – symbole pozwalają grupować standardowe wyjścia poleceń np. w celu ich przekierowania do pliku. Przykładowo ich użycie:

```

1 (cat /etc/passwd | cut -d: -f3; ls -al; man ps) > dane.txt

```

spowoduje zapisanie do pliku *dane.txt* numerów identyfikacyjnych użytkowników zdefiniowanych w lokalnej bazie, zawartości bieżącego katalogu oraz stron pomocy dla polecenia *ps*.

**\$zmienna** – znak \$ należy traktować jak operator pozwalający pobrać wartość zmiennej, krócej nazwa pojawiła się po nim. Nazwa zmiennej musi być dołączona do znaku. Nawiasy { } służą do zaznaczenia zakresu nazwy zmiennej. We współczesnych interpreterach często się je pomija, gdyż zakładają one, że nazwa zmiennej zaczyna się pierwszym znakiem po znaku \$ i kończy na ostatnim znaku przed znakiem białym lub znakiem specjalnym.

‘polecenie’ – symbole odwróconych apostrofów pozwalają na wykonanie polecenia w nich zawartego polecenia i podstawienie znaków wygenerowanych przez nie na standardowe wyjście do wiersza poleceń. Nie jest możliwe zagnieżdżanie czyli wykonywanie polecenia w poleceniu. Przykładowo polecenie:

```
1 ls -l ‘which passwd’
```

spowoduje wypisanie w postaci długiej informacji o pliku zawierającym kod polecenia *passwd*, gdyż najpierw wykonane zostanie polecenie *which passwd*, które zwróci bezwzględną ścieżkę dostępu do tegoż pliku.

\ – w zależności od miejsca, w którym pojawi się w wierszu polecenia, posiada trzy znaczenia:

1. Na początku, przed poleceniem do wykonania nakazuje interpreterowi poszukiwanie polecenia jako zewnętrznego (z pominięciem poleceń wbudowanych oraz aliasów).
2. Na końcu wierszu polecenia nakazuje kontynuowanie polecenia w kolejnej linii. Pojawia się tzw. drugi znak zachęty (lub kontynuacji). Naciśnięcie klawisza Enter w linii, która nie kończy się znakiem \ powoduje złożenie polecenia z linii składowych i przejście do kolejnych etapów jego wykonywania. Złożenie jest zależne od rodziny interpretera. Interpreter *bash* składając polecenie z kolejnych linii usuwa znak \. Stąd należy pamiętać o umieszczaniu białych znaków w odpowiednich miejscach. Dzielenie wiersza polecenia jest zatem możliwe w dowolnym miejscu. Interpretery z rodziny *csh* (*csh*, *tcsh*) zastępują znak \ znakiem spacji. Stąd dzielenie polecenia nie jest możliwe w dowolnych miejscach, a jedynie tu, gdzie wymagany jest biały znak.
3. W środku wiersza polecenia służy do zasłaniania specjalnego znaczenia jednego znaku, który po nim występuje.

”napis” – cudzysłowia służą do cytowania znaków, które między nimi występują. W wierszu polecenia muszą one być sparowane. Shell interpretuje w napisie, który się między nimi pojawił, znaki rozwinięcia nazw plików i katalogów, znak \$ oraz odwrócone apostrofy.

’napis’ – apostrofy, podobnie jak cudzysłowia, służą do cytowania zawartych między nimi znaków. W wierszu polecenia muszą być sparowane. Shell nie interpretuje żadnych znaków specjalnych między nimi występujących.

Na zakończenie jedna uwaga. Otóż znaki rozwijania nazw plików i katalogów są często mylone ze znakami stosowanymi w wyrażeniach regularnych, np. w poleceniu *grep*.

## 5.3 Konfiguracja

Interpreter poleceń, z punktu widzenia programisty, jest zwykłym programem. Na sposób jego działania wpływ ma jedynie występowanie lub występowanie i wartości zmiennych o zdefiniowanych nazwach. Wartości zmiennych są domyślne, czyli przypisane w programie interpretera, albo definiowane przez użytkownika. Zdefiniowanie wartości przez użytkownika zastępuje wartość domyślną, jeśli ta została nadana. Wartość zmiennej może być napisem lub liczbą. W interpreterach poleceń wyróżnia się zasadniczo trzy rodzaje zmiennych. Są to:

1. Zmienne specjalne.
2. Zmienne środowiska.

### 3. Zmienne programowe.

Kolejne podrozdziały zawierają krótkie omówienie podstawowych zmiennych każdego rodzaju.

#### 5.3.1 Zmienne specjalne

#### 5.3.2 Zmienne środowiska

#### 5.3.3 Zmienne programowe

## 5.4 Programowanie w językach interpreterów poleceń

Konieczność automatyzacji niektórych czynności, zarówno administracyjnych jak i wykonywanych przez zwykłego użytkownika, prowadząca do podniesienia efektywności pracy spowodowały wyposażenie interpreterów poleceń w języki programowania. Są one pełnymi językami programowania. Posiadają instrukcje warunkowe i iteracyjne, a ich cechą charakterystyczną jest łatwość wykorzystywania poleceń zewnętrznych. Podnosi to znacznie efektywność programowania, gdyż polecenia te dostarczają duży zbiór gotowych funkcjonalności.

Na wstępie zostanie omówione polecenie *expr* umożliwiające obliczanie wartości wyrażeń. W instrukcjach warunkowych konieczna jest umiejętność porównywania wartości oraz sprawdzania typów plików. Stąd krótkie omówienie polecenia *test*. Zarówno *expr*, jak i *test* są poleceniami zewnętrznymi, stąd możliwość ich wykorzystania w skrypcie napisanym dla dowolnego interpretera poleceń. W dalszej części przedstawione zostaną sposoby implementacji i uruchamiania skryptów. Na koniec kilka przykładowych skryptów, od najprostszych do nieco bardziej złożonych.

### 5.4.1 Polecenie *expr*

Polecenie *expr* posiada następującą składnię:

*expr* **opcja**

*expr* **wyrażenie**

W pierwszym przypadku możliwe jest użycie opcji – **–help** dostarczającej ogólnych informacji o poleceniu oraz – **–version**, powodującej wypisanie jego wersji. Uruchomienie polecenia z argumentem będącym wyrażeniem spowoduje wyznaczenie jego wartości i wypisanie jej na standardowe wyjście. Podstawowe wyrażenia arytmetyczne mogą mieć postać:

ARG1 \* ARG2 – iloczyn wartości argumentów.

ARG1 / ARG2 – iloraz wartości argumentów.

ARG1 % ARG2 – reszta z dzielenia wartości ARG1 przez ARG2.

ARG1 + ARG2 – suma wartości argumentów.

ARG1 – ARG2 – różnica wartości argumentów.

Operatory zostały przytoczone w kolejności priorytetów od najwyższego do najniższego. Generalnie polecenie *expr* przyjmuje dwa argumenty. Korzystając z nawiasów ( oraz ) można budować dowolne wyrażenia oraz zmieniać kolejność operacji wynikającą z priorytetów operatorów. Należy jedynie pamiętać o użyciu znaku odwróconego ukośnika (\) przez nawiasem, gdyż jest on znakiem specjalnym dla interpretera poleceń. Przykłady:

```

1 [jan@messy ~]$ expr \( 2 + 2 \)
2 4
3 [jan@messy ~]$ expr \( 2 + 2 \) \* 2
4 8
5 [jan@messy ~]$ expr 2 + \( 2 \* 2 \)
6 6
7 [jan@messy ~]$ expr \( 2 + \( 2 \* 2 \) \) + 4
8 10

```

Polecenie *expr* Służy również do porównywania wartości. Zasadniczo zwraca ono wartość 1 jeśli wynik porównania jest prawdą lub wartość 0 jeśli jest fałszem. Może również zwracać wartość jednego z argumentów. Podstawowe operatory to:

$ARG1 < ARG2$  - zwraca wartość 1 jeśli wartość argumentu ARG1 jest mniejsza od wartości argumentu ARG2.

$ARG1 \leq ARG2$  - zwraca zwraca wartość 1 jeśli wartość argumentu ARG1 jest mniejsza lub równa od wartości argumentu ARG2.

$ARG1 = ARG2$  - zwraca zwraca wartość 1 jeśli wartość argumentu ARG1 jest równa wartości argumentu ARG2.

$ARG1 \neq ARG2$  - zwraca zwraca wartość 1 jeśli wartość argumentu ARG1 jest różna od wartości argumentu ARG2.

$ARG1 > ARG2$  - zwraca zwraca wartość 1 jeśli wartość argumentu ARG1 jest większa od wartości argumentu ARG2.

$ARG1 \geq ARG2$  - zwraca zwraca wartość 1 jeśli wartość argumentu ARG1 jest większa lub równa od wartości argumentu ARG2.

$ARG1 \parallel ARG2$  - zwraca wartość argumentu ARG1 jeśli jest ona różna od zera, w przeciwnym przypadku wartość argumentu ARG2.

$ARG1 \& ARG2$  - zwraca wartość argumentu ARG1 jeśli wartości obu argumentów są różne od zera, zaś zero w przypadku przeciwnym.

Przykłady zamieszczono poniżej. Jeśli operator polecenia *expr* jest znakiem specjalnym dla interpretera poleceń, to wówczas należy go zabezpieczyć przez interpretacją biorąc go w cudzysłowia, apostrofy lub stawiając przed nim odwrócony ukośnik (linie 5, 7, 12, 15, 18, 21, 24 oraz 27).

```

1 [jan@messy ~]$ L1=10
2 [jan@messy ~]$ L2=17
3 [jan@messy ~]$ expr $L1 = $L2
4 0
5 [jan@messy ~]$ expr $L1 \> $L2
6 0
7 [jan@messy ~]$ expr $L1 \< $L2
8 1
9 [jan@messy ~]$ expr $L1 != $L2

```

```

10 1
11 [jan@messy ~]$ A1=10; A2=37
12 [jan@messy ~]$ expr $A1 \|| $A2
13 10
14 [jan@messy ~]$ A1=0
15 [jan@messy ~]$ expr $A1 \|| $A2
16 37
17 [jan@messy ~]$ A1=10; A2=0
18 [jan@messy ~]$ expr $A1 \|| $A2
19 10
20 [jan@messy ~]$ A1=10; A2=27
21 [jan@messy ~]$ expr $A1 \& $A2
22 10
23 [jan@messy ~]$ A1=0
24 [jan@messy ~]$ expr $A1 \& $A2
25 0
26 [jan@messy ~]$ A1=10; A2=0
27 [jan@messy ~]$ expr $A1 \& $A2
28 0

```

Bardzo cenne podczas pisanie programów są operacje na napisach. Podstawowe możliwości polecenia *expr* w tym zakresie są następujące:

*substr* NAPIS POZYCJA DŁUGOŚĆ - wypisuje na standardowe wyjście fragment napisu NAPIS od znaku znajdującego się na pozycji POZYCJA o długości DŁUGOŚĆ. Numer pozycji pierwszego znaku to 1.

*index* NAPIS ZNAK - wypisuje na standardowe wyjście pozycje wystąpienia znaku ZNAK w napisie NAPIS lub 0 jeśli znak nie znajduje się w napisie.

*length* NAPIS - wypisuje długość napisu.

*match* NAPIS WYRAŻENIE\_REGULARNE (lub NAPIS : WYRAŻENIE\_REGULARNE) - wypisuje na standardowe wyjście długość napisu opisanego wyrażeniem regularnym znajdującym się w napisie NAPIS lub wartość 0 jeśli nie znaleziono.

Kilka prostych przykładów zamieszczono poniżej. Należy zwrócić uwagę, że jeżeli napis zawiera znaki białe, to wówczas musi on pojawić się w cudzysłowach (linie 2, 4, 6, 8). Dodatkowo wyrażenie regularne, które zawiera znaki specjalne interpretera poleceń musi zostać zapisane w apostrofach lub cudzysłowach.

```

1 [jan@messy ~]$ NAPIS='Po prostu napis testowy'
2 [jan@messy ~]$ expr substr "$NAPIS" 4 6
3 prostu
4 [jan@messy ~]$ expr index "$NAPIS" s
5 7
6 [jan@messy ~]$ expr length "$NAPIS"
7 23
8 [jan@messy ~]$ expr match "$NAPIS" ".* pro"
9 6

```

### 5.4.2 Polecenie *test*

Polecenie *test* służy do porównywania wartości oraz sprawdzania typów plików. W wyniku działania zwraca kod powrotu określony wartością wyrażenia. Jeśli wyrażenie jest prawdziwe to zwracana jest wartość 0 (sukces). W przypadku przeciwnym kod powrotu ma wartość 1 (błąd wykonania). Polecenie posiada jedynie 2 opcje pozwalające na wypisanie krótkiej informacji o składni polecenia i sposobach jego użycia (**-help** oraz wersji polecenia (**-version**). Uruchomienie polecenia może przyjąć jedną z następujących postaci:

*test* **WYRAŻENIE**

[ **WYRAŻENIE** ]

Zacznijmy od opisu wyrażeń porównujących wartości zmiennych. Dla liczb całkowitych możliwe są:

LICZBA\_1 **-eq** LICZBA\_2 - zwraca prawdę, jeśli wartości są równe.

LICZBA\_1 **-ge** LICZBA\_2 - zwraca prawdę, jeśli wartość LICZBA\_1 jest większa lub równa od wartości LICZBA\_2.

LICZBA\_1 **-gt** LICZBA\_2 - zwraca prawdę, jeśli wartość LICZBA\_1 jest większa od wartości LICZBA\_2.

LICZBA\_1 **-le** LICZBA\_2 - zwraca prawdę, jeśli wartość LICZBA\_1 jest mniejsza lub równa od wartości LICZBA\_2.

LICZBA\_1 **-lt** LICZBA\_2 - zwraca prawdę, jeśli wartość LICZBA\_1 jest mniejsza od wartości LICZBA\_2.

LICZBA\_1 **-ne** LICZBA\_2 - zwraca prawdę, jeśli wartości są różne.

Poniżej zamieszczono kilka prostych przykładów:

```

1  [jan@messy ~] L1=10; L2=20
2  [jan@messy ~] [ $L1 -eq $L2 ]
3  [jan@messy ~] echo $?
4  1
5  [jan@messy ~] [ $L1 -gt $L2 ]
6  [jan@messy ~] echo $?
7  1
8  [jan@messy ~] [ $L1 -le $L2 ]
9  [jan@messy ~] echo $?
10 0
11 [jan@messy ~] [ $L1 -ne $L2 ]
12 [jan@messy ~] echo $?
13 0

```

Możliwe jest także badanie napisów w następującym zakresie:

**NAPIS** (lub **-n NAPIS**) - zwraca prawdę jeśli napis jest napisem niepustym.

**-z NAPIS** - wyrażenie jest prawdziwe jeśli badany napis jest pusty,

`NAPIS_1 = NAPIS_2` - jeśli napisy `NAPIS_1` i `NAPIS_2` są identyczne to wyrażenie jest prawdziwe.

`NAPIS_1 != NAPIS_2` - zwracana jest wartość prawdy jeśli napisy `NAPIS_1` i `NAPIS_2` nie są identyczne.

Przykłady użycia zamieszczono poniżej. Należy zwrócić uwagę na umieszczanie wartości zmiennych w apostrofach lub cudzysłowach, gdyż mogą one zawierać znaki spacji lub tabulacji.

```

1 [jan@messy ~] N1="Jan Kowalski"; N2="Antoni Nowak"
2 [jan@messy ~] [ "$N1" ]
3 [jan@messy ~] echo $?
4 0
5 [jan@messy ~] [ -z "$N1" ]
6 [jan@messy ~] echo $?
7 1
8 [jan@messy ~] [ "$N1" != "$N2" ]
9 [jan@messy ~] echo $?
10 0

```

Polecenie `test` posiada również możliwości pozwalające na porównywanie wartości wybranych atrybutów plików oraz określanie wartości atrybutu pliku. Przytoczone poniżej wyrażenia zwracają prawdę, jeśli:

`PLIK_1 -ef PLIK_2` - oba pliki mają to samo urządzenie oraz numery i-węzłów.

`PLIK_1 -nt PLIK_2` - `PLIK_1` był później modyfikowany (jest nowszy) niż `PLIK_2`.

`PLIK_1 -ot PLIK_2` - `PLIK_1` był wcześniej modyfikowany (jest starszy) niż `PLIK_2`.

`-b PLIK` - plik istnieje i reprezentuje urządzenie blokowe.

`-c PLIK` - plik istnieje i reprezentuje urządzenie znakowe.

`-d PLIK` - plik istnieje i jest katalogiem.

`-e PLIK` - plik istnieje.

`-f PLIK` - plik istnieje i jest plikiem regularnym.

`-g PLIK` - plik istnieje i ma nadane prawo SGID.

`-h PLIK` - plik istnieje i jest dowiązaniem symbolicznym (analogicznie do `-L`).

`-G PLIK` - plik istnieje i jest własnością efektywnej grupy.

`-k PLIK` - plik istnieje i ma nadane prawo SVTX (bit lepki).

`-L PLIK` - plik istnieje i jest dowiązaniem symbolicznym (analogicznie do `-h`).

`-O PLIK` - plik istnieje i jest własnością efektywnego użytkownika.

`-p PLIK` - plik istnieje i jest typu łącza nazwanego (ang. *named pipe*).

`-r PLIK` - plik istnieje i użytkownik, który uruchomił polecenie `test` ma prawo do odczytu.

- s – plik istnieje i ma rozmiar większy od zera.
- S – plik istnieje i jest typu socket.
- t DESKRYPTOR – deskryptor pliku (domyślnie standardowe wyjście) jest otwarty na terminalu.
- u PLIK – plik istnieje i ma nadane prawo SUID.
- w PLIK – plik istnieje i użytkownik, który uruchomił polecenie test ma prawo do zapisu.
- x PLIK – plik istnieje i użytkownik, który uruchomił polecenie test ma prawo do wykonywania.

### 5.4.3 Zasady uruchamiania skryptów

Zacznijmy jednak od omówienia sposobów implementacji i uruchamiania skryptów. W najprostszym przypadku skrypt możemy zapisać w wierszu polecenia. Przykład skryptu, który w pętli nieskończonej wyznacza liczbę

### 5.4.4 Przykładowe, proste skrypty

Poniżej zamieszczono przykłady kilku prostych skryptów. Stanowią one jedno z wielu możliwych rozwiązań postawionego zadania. Zapewne nie uwzględniają wszystkich, możliwych sytuacji, które mogą pojawić się podczas ich wykonania w rzeczywistym systemie. Zadaniem przed nimi postawionym jest zrozumienie filozofii programowania dla interpreterów poleceń oraz nabywanie biegłości praktycznego wykorzystania podstawowych instrukcji skryptowych języków programowania.

**Skrypt1** oblicza i wypisuje na ekran liczbę sekund, które procesor poświęcił na wykonanie instrukcji wszystkich procesów, których właścicielem efektywnym jest bieżący użytkownik.

```

1  #!/bin/bash
2
3  CZASY='ps aux | awk '{print $1":"$10}' | grep "^$USER:" | \
4      awk -F: '{print $2*60+$3}'
5  RAZEM=0
6  for CZAS in $CZASY; do
7      RAZEM='expr $RAZEM + $CZAS'
8  done
9  echo "Procesy użytkownika $USER zużyły $RAZEM [s] czasu procesora"
```

**Skrypt2** musi zostać uruchomiony z dwoma argumentami, którymi są zakończenia nazw plików poprzedzone znakiem „.”. Skrypt dokonuje kopiowania wszystkich plików regularnych z katalogu bieżącego, których nazwy kończą się „.” oraz ciągiem znaków będącym pierwszym argumentem do plików, których nazwa to ciąg znaków będący fragmentem nazwy pliku oryginalnego do pierwszej kropki włącznie połączonej z drugim argumentem (przykładowo uruchomienie `skrypt1.jpg gif` spowoduje skopiowanie zawartości wszystkich plików z bieżącego o nazwach zakończonych na `.jpg` do plików o nazwach zakończonych na `.gif`, np. zawartość pliku o nazwie `obraz.jpg`



ma zostać skopiowana do pliku o nazwie `obraz.gif`). Skrypt wypisuje również informację o liczbie skopiowanych plików. Uruchomienie z inną liczbą argumentów powoduje jedynie wypisanie informacji o sposobie uruchomienia.

```

1  #!/bin/bash
2  if [ $# -ne 2 ]; then
3      echo "Uzycie: $0 napis1 napis2"
4      exit 1
5  else
6      RAZEM=0
7      for PLIK in *.* $1* *.* $1*; do
8          if [ -f $PLIK ]; then
9              NAZWA='echo $PLIK | awk -F '.' '{print $1}''
10             NAZWA=$NAZWA'.'"$2"
11             cp $PLIK $NAZWA
12             RAZEM='expr $RAZEM + 1'
13         fi
14     done
15     echo "Liczba skopiowano plikow: $RAZEM"
16 fi

```

**Skrypt3** może zostać wywołany z jednym argumentem, będącym ścieżką dostępu do katalogu. W takim przypadku, dla dalszego działania musi sprawdzić, czy katalog ten istnieje. Jeśli katalog istnieje, to skrypt ma obliczyć ile w sumie bajtów mają znajdujące się w nim wszystkie pliki regularne i wypisać obliczoną wartość na standardowe wyjście. Jeśli skrypt został wywołany bez argumentu lub z liczbą argumenów większą niż jeden, to wówczas obliczenie sumy ma dotyczyć katalogu bieżącego.

```

1  #!/bin/bash
2  KAT_BIEZACY='pwd'
3  if [ $# -eq 1 ]; then
4      if [ -d $1 ]; then
5          cd $1
6      else
7          echo "Katalog $1 nie istnieje"
8          exit 1
9      fi
10 fi
11 RAZEM=0
12 KAT_ROBOCZY='pwd'
13 for PLIK in .* *; do
14     if [ -f $PLIK ]; then
15         ROZMIAR='ls -l $PLIK | awk '{print $5}''
16         RAZEM='expr $RAZEM + $ROZMIAR'
17     fi
18 done
19 echo "Rozmiar plikow regularnych w katalogu $KAT_ROBOCZY wynosi $RAZEM bajtow"
20 cd $KAT_BIEZACY

```

**Skrypt4** przyjmuje jako argument liczbę naturalną, która oznacza maksymalną liczbę sekund czasu procesora poświęconych na wykonanie instrukcji danego procesu. Jeśli skrypt został uruchomiony bez argumentu, to przyjmuje wartość domyślną wynoszącą 30 sekund. Skrypt zmienia wartość parametru NICE każdego procesu, który przekroczył graniczną wartość zużycia czasu procesora na 5 oraz informuje ilu procesom wartość ta została zmieniona.

```

1  #!/bin/bash
2
3  if [ $# -eq 0 ]; then
4      echo -n "Graniczna liczba sekund: "
5      read S_MAX
6  else
7      S_MAX=$1
8  fi
9  LISTA='ps -el | tail -n+2 | awk '{print $4":"$8":"$13}''
10 for PROCESS in $LISTA; do
11     LISTA_PROC=$LISTA_PROC'echo $PROCESS | awk -v wart=$S_MAX -F: \
12         '{ if ($3*3600+$4*60+$5 > wart && $2<5) print $1" " }','
13 done
14 RAZEM=0
15 for PROC_ID in $LISTA_PROC; do
16     renice 5 -p $PROC_ID 2>/dev/null
17     if [ $? -eq 0 ]; then
18         RAZEM='expr $RAZEM + 1'
19     fi
20 done
21 echo "Liczba procesow ktorym zmieniono wartosc parametru NICE: $RAZEM"

```

**Skrypt5** może zostać uruchomiony z co najmniej jednym argumentem, którym jest nazwa użytkownika w systemie. Jeśli skrypt został uruchomiony bez argumentu, to musi zapytać o nazwę (nazwy) użytkowników. Skrypt oblicza ile pamięci rezydualnej zajmują łącznie procesy użytkownika (użytkowników – każdego z osobna) o podanych nazwach.

```

1  #!/bin/bash
2  if [ $# -eq 0 ]; then
3      echo -n "Podaj nazwy uzytkownika(ow) oddzielone spacja: "
4      read UZYTKOWNICY
5  else
6      UZYTKOWNICY=$*
7  fi
8  for UZYTEK in $UZYTKOWNICY; do
9      ROZMIARY='ps aux | tail -n+2 | awk '{print $1":"$6}''\
10         | grep "^$UZYTEK:" | awk -F: '{print $2}''
11      RAZEM=0
12      for I in $ROZMIARY; do
13          RAZEM=$((RAZEM + $I)) # zamiast RAZEM='expr $RAZEM + $I'
14      done
15      echo "    $UZYTEK    $RAZEM"
16  done

```

**Skrypt6** pracując w pętli nieskończonej, co 5 sekund wypisuje listę zalogowanych użytkowników z podaniem liczby procesów, których właścicielem efektywnym jest dany użytkownik. Przerwanie pracy skryptu następuje przez naciśnięcie klawiszy *Ctrl + C*, po czym skrypt ma wypisać bieżącą datę i godzinę.

```

1  #!/bin/bash
2
3  trap 'DATA='date'; echo "Przerwanie Ctrl+C"; echo "$DATA"; exit 0' 2
4
5  while [ 1 ]; do
6      echo "Uzytkownik | liczba procesow"
7      echo "-----"
8      UZYTKOWNICY='who | awk '{ print $1 }' | sort | uniq'
9      for UZYTEK in $UZYTKOWNICY; do
10         LPROC='ps -eo "%U" | grep -c $UZYTEK'
11         echo " $UZYTEK      $LPROC"
12     done
13     echo "-----"
14     sleep 3
15 done

```

### 5.4.5 Przykładowe skrypty

Usuwanie użytkownika z systemu

Reakcja na problemy w działaniu macierzy RAID

```

1  #!/bin/sh
2  # MDADM Event Handler - Generate mails when MD events occur
3
4  CONFIG="/etc/mdadm.conf"
5
6  # Overrule mailto address from the mdadm.conf file?:
7  #MAILADDR="root"
8
9  parse_event()
10 {
11     echo "Host          : $HOSTNAME"
12     if [ -z "$1" ]; then
13         echo "Event          : Test message"
14     else
15         echo "MD Device        : $2"
16         echo "Event          : $1"
17         if [ -n "$3" ]; then
18             echo "Device Component: $3"
19         fi
20     fi
21
22     echo ""
23     echo "/proc/mdstat dump:"

```

```

24 FAIL=0
25 DEGRADED=0
26 while read LINE; do
27     echo -n "$LINE"
28     if [ -n "$(echo "$LINE" |grep 'active raid')" ]; then
29         if [ -n "$(echo "$LINE" |grep '\(F\)')" ]; then
30             FAIL=$((FAIL + 1))
31             echo -n " (WARNING: FAILED DISK(S)!)"
32         fi
33
34         if [ -n "$(echo "$LINE" |grep '\(S\)')" ]; then
35             echo -n " (Hotspare(s) available)"
36         else
37             echo -n " (NOTE: No hotspare?!)"
38         fi
39     fi
40
41     if [ -n "$(echo "$LINE" |grep 'blocks')" ]; then
42         if [ -n "$(echo "$LINE" |grep '_')" ]; then
43             DEGRADED=$((DEGRADED + 1))
44             echo -n " (DEGRADED!!)"
45         fi
46     fi
47
48     echo ""
49 done < /proc/mdstat
50
51 if [ $FAIL -gt 0 ]; then
52     echo ""
53     echo "** WARNING: One or more RAID arrays have FAILED disk(s)! **"
54 fi
55
56 if [ $DEGRADED -gt 0 ]; then
57     echo ""
58     echo "** WARNING: One or more RAID arrays are running in degraded mode! **"
59 fi
60 }
61
62 # main line:
63 # Get MAILADDR from mdadm.conf config file, if not set already
64 if [ -z "$MAILADDR" ] && [ -f "$CONFIG" ]; then
65     MAILADDR='grep MAILADDR "$CONFIG" |cut -d' ' -f2'
66     if [ -z "$MAILADDR" ]; then
67         MAILADDR="root"
68     fi
69 fi
70
71 # Call the parser and send it to the configured address
72 parse_event $* |mail -s "RAID(MD) event on $HOSTNAME" "$MAILADDR"

```

73

74

`exit 0`

## Rozdział 6

# Zarządzanie oprogramowaniem

## Rozdział 7

# Podstawy konfiguracji sieciowych

## Rozdział 8

# Badanie wydajności systemu i optymalizacja



# Bibliografia

- [1] Silberschatz A. and Galvin P. *Podstawy systemów operacyjnych*. Wydawnictwa Naukowo–Techniczne, Warszawa, 2000.
- [2] Organick E. J. *The Multics System: An Examination of Its Structure*. MIT, Cambridge, Ma, 1972.
- [3] Thompson K. and Ritchie D. The unix time-sharing system. *Communications of the ACM*, 17:365–375, 1974.
- [4] David F. Ferraiolo D. Richard Kuhn and Ramaswamy Chandramouli. *Role-Based Access Control*. Artech House Publishers, Norwood, 2003.
- [5] Marshall K. McKusick, William N. Joy, Samuel J. Leffler, and Robert S. Fabry. A fast file system for unix. *ACM Trans. Comput. Syst.*, 2(3):181–197, 1984.
- [6] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). *SIGMOD Rec.*, 17(3):109–116, 1988.
- [7] Sandhu R.S., E.J. Coyne, Feinstein H.L., and Youman C.E. Role-based access control models. *Computer*, 29:38–47, 1996.
- [8] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive, Report 2004/199, 2004. <http://eprint.iacr.org/>.

# Indeks

a.out, 375  
adduser, 53  
attr, 217, 233  
audit, 103  
  
badblocks, 220, 255  
btrfs, 290  
btrfs-convert, 288, 290  
btrfs-debug-tree, 290  
btrfs-find-root, 290  
btrfs-image, 290  
btrfs-map-logical, 291  
btrfs-restore, 291  
btrfs-show, 291  
btrfs-vol, 291  
btrfs-zero-log, 291  
btrfsctl, 290  
btrfstune, 291  
  
cap\_mkdb, FreeBSD, 97  
chacl, 128, 233  
chage, 22  
chattr, 198, 202, 203  
chfn, 22, 53  
chgrp, 134  
chmod, 122  
chown, 46, 134  
chpass, 53  
chsh, 22, 53  
  
debugfs, 202, 203  
debugfs.reiser4, 275  
debugreiserfs, 269  
df, 115  
dump, 146, 261  
dumpfs, 307, 310  
  
e2fsck, 203, 205  
e2label, 146, 215  
edquota, 164, 169, 178, 182  
ELF, 375  
  
exec, 375  
exit, 376  
expr, 395  
  
faillog, 91  
fdisk, 190, 293  
findsuper, 204  
fork, 375  
fsck, 147, 310  
fsck.ext4, 219  
fsck.reiser4, 265  
fsck.xfs, 232  
fsck\_ffs, 310  
fsck\_ufs, 310  
fsdb, 310  
fuser, 156  
  
geom, 369  
getfacl, 125, 130, 233  
gjournal, 305  
grep, 394  
grpck, 47  
  
jfs\_debugfs, 244, 245  
jfs\_fscklog, 246  
jfs\_logdump, 246  
jfs\_tune, 247  
  
last, 32  
ln, 114  
lsattr, 204  
lsof, 159  
lvcreate, 329  
lvdisplay, 329  
lvextend, 273, 332  
lvresize, 273, 332  
  
make\_reiser4, 255  
mdadm, 353  
measurefs.reiser4, 277  
mesg, Fedora, 33

- mke2fs, 204
- mkfifo, 113
- mkfs, 199
- mkfs.ext3, 199
- mkfs.jfs, 239
- mkfs.xfs, 232
- mknod, 112, 113
- mkreiserfs, 253
- mount, 148, 152
  
- newfs, 304
- newsyslog, FreeBSD, 71
  
- parted, 190
- partprobe, 195
- passwd, 53
- passwd, komenda, 21
- praudit, 105
- prioctl, 382
- pvccreate, 327
- pvmove, 336
- pw, 53
- pwck, 47
- pwd\_mkdb, 51
  
- quota, 163, 171, 177, 183
- quotacheck, 163, 166, 177, 181
- quotaoff, 163, 168, 177
- quotaon, 163, 168, 177, 182
  
- RAID-Z, 317
- reiserfsck, 261
- reiserfstune, 263, 272
- repquota, 165, 172, 179, 184
- repquote, 184
- rescuept, 205
- resize2fs, 204, 332
- resize\_reiserfs, 261, 273
- rmuser, 53
  
- SAMBA, 155
- setfacl, 127, 132, 233
- setquota, 170
- sfdisk, 190
- sg, 134
- signal, 377
- sync, 185
- sysinstall, 293
- syslogd, FreeBSD, 386
  
- test, 398
  
- tune2fs, 146, 200, 204, 207, 214
- tunefs, 308
  
- ulimit, 95
- umask, 121
- umount, 156
- uptime, BSD, 69
- useradd, 21
- userdel, 22
- usermod, 22
- uuidgen, 255
  
- vgchange, 327, 340
- vgcreate, 327
- vgdisplay, 327
- vgexport, 340
- vgextend, 334
- vgimport, 340
- vgreduce, 337
- vgscan, 325
- vipw, 51
  
- w, 32
- wait, 376
- warnquota, 166, 173, 185
- who, 32
  
- xfs\_admin, 232
- xfs\_bmap, 232
- xfs\_check, 229, 232
- xfs\_copy, 233
- xfs\_db, 229, 232
- xfs\_estimate, 233
- xfs\_freeze, 232
- xfs\_fsr, 233
- xfs\_growfs, 232
- xfs\_info, 232
- xfs\_logprint, 232
- xfs\_ncheck, 232
- xfs\_repair, 229, 232
- xfs\_rtcp, 232
- xfsdq, 233
- xfsdump, 233
- xfsinvutil, 233
- xfsrestore, 233
- xfsrq, 233
  
- zpool, 318