Kurs administrowania systemem Linux

Zajęcia nr 15: Systemy plików, cz. 2

Instytut Informatyki Uniwersytetu Wrocławskiego

10 czerwca 2024

Złożoność systemów plików

System	Liczba osobolat
ext4	8.5
XFS	17
ZFS	77

Złożoność systemów plików

System	Liczba osobolat
ext4	8.5
XFS	17
ZFS	77

I wrote FAT on an airplane, for heaven's sake.

Bill Gates (as quoted by Raymond Chen, 2013)

FAT

Struktura systemu na dysku

- Region zarezerwowany
- Tablica FAT
- Katalog główny (nie dotyczy FAT32)
- Region plików i katalogów

Przydział pamięci

- Jednostka adresowania: *klaster*. Numery klastrów są 8- (oryginalny system DOS), 12- (FAT12), 16- (FAT16) lub 28-bitowe (FAT32, zapisywane w 4 bajtach, 4 najbardziej znaczące bity zarezerwowane).
- Jeśli liczba klastrów jest mniejsza niż 4085, to mamy FAT12, jeśli większa lub równa 4085 i mniejsza od 65525, to FAT16, w p.p. FAT32.
- Rozmiar klastra: 1, 2, 4, 8, 16, 32, 64 lub 128 sektorów.
- Najmniejszy legalny rozmiar FAT32: 65525+1 sektorów.
- Maksymalna liczba klastrów w FAT32: $2^{28} - 2 = 268435454$. Dla klastrów 16-sektorowych rozmiar wolumenu ok. 2TB.

BIOS Parameter Block (Boot sector: MBR lub PBR)

Offset	Rozmiar	Zawartość
0	3	0xEB*XX*90 (`jmp` *XX*)
3	8	Nazwa kreatora (oryg. `MSWIN4.1`)
11	2	Rozmiar sektora w B
13	1	Liczba sektorów na klaster
14	2	Rozmiar regionu zarezerwowanego w sektorach
16	1	Liczba tablic FAT (zwykle 2)
17	2	Liczba wpisów w katalogu głównym (FAT12 i 16)
19	2	Rozmiar wolumenu w sektorach (FAT12 i 16)
21	1	Typ nośnika
22	2	Rozmiar tablicy FAT w sektorach (FAT12 i 16)
24	2	Liczba sektorów na ścieżkę
26	2	Liczba głowic
28	4	Liczba ukrytych sektorów
32	4	Rozmiar wolumenu w sektorach

BPB, cd. dla FAT32

Offset	Rozmiar	Zawartość
36	4	Rozmiar tablicy FAT w sektorach
40	2	Konfiguracja tablic FAT
42	2	Wersja systemu plików
44	4	Adres katalogu głównego (zwykle 2 — pierwszy klaster)
48	2	FSINFO
50	2	Adres (w sektorach) backupu BPB
52	12	Zarezerwowane
64	1	Numer napędu
65	1	Zarezerwowane, wpisać 0
66	1	Rozszerzona sygntura 0x29
67	4	ID wolumenu
71	11	Etykieta wolumenu
82	8	Zawsze napis `FAT32\x20\x20\x20`

Tablica FAT

- Dla każdego klastra określa następny klaster w łańcuchu.
- Specjalne numery klastrów w FAT32: `0` — nieużywany, `0x0ffffff7` — uszkodzony, `0x0fffffff` — koniec łańcucha klastrów.
- Indeksy i wartości — numery klastrów.
- Zużyty obszar: *liczba klastrów* $\times$ *rozmiar numeru klastra*.
- Ułamek: *rozmiar numeru klastra*/*rozmiar klastra*, np. $4/8192 = 1/2048$ dla FAT32 i klastrów 16-sektorowych.
- Przykład: pendrive 16GB, klastry 8192 B: 2'097'152 klastrów. Rozmiar FAT: 8 MB.
- Problem: wyznaczenie liczby wolnych klastrów wymaga przejrzenia całej tablicy FAT (są rozszerzenia w FAT32).

Katalog w FAT32

Offset	Rozmiar	Zawartość
0	11	Nazwa (8+3)
11	1	Atrybuty (6 bitów)
12	1	Zarezerwowane, wpisać 0
13	1	1/10 sekundy czasu utworzenia (0–199)
14	2	Czas utworzenia
16	2	Data utworzenia
18	2	Data ostatniego dostępu
20	2	Dwa górne bajty numeru klastra
22	2	Czas ostatniej modyfikacji
24	2	Data ostatniej modyfikacji
26	2	Dwa dolne bajty numeru klastra
28	4	Rozmiar pliku w bajtach

Uwagi

- Rozdzielczość zapisu czasu: 2 sekundy.
- Maksymalny rozmiar pliku: $2^{32} - 1\,\mathrm{B}$, czyli 4 GB.

Nazwa pliku

Krótkie nazwy

- Małe i wielkie litery utożsamiane (są tylko wielkie).
- Format: 8+3, uzupełniane z prawej spacjami.
- Pierwszy znak `0xE5` — wolny wiersz w katalogu.
- Pierwszy znak `0x00` — wolny wiersz w katalogu, brak dalszych.
- Pierwszy znak `0x05` — znak `0xE5`.

Długie nazwy

- Konserwatywne rozszerzenie.
- Dodatkowe wpisy w katalogu.

Format nie definiuje procedur

Design space

- Wybór typu systemu (16–32) i jego parametrów (rozmiar klastra).
- Polityka alokacji klastrów — czy można zmniejszyć fragmentację?
- Prealokacja
- Proces defragmentacji
- Spójność wpisów w katalogach (problem długich nazw)

Naginanie formatu — czy można?

- *hard links*
- deduplikacja

Microsoft FAT32 File System Specification

The “fixed value” is simply a volume size that is the “FAT16 to FAT32 cutover value”. Any volume size smaller than this is FAT16 and any volume of this size or larger is FAT32. For Windows, this value is 512 MB. Any FAT volume smaller than 512 MB is FAT16, and any FAT volume of 512 MB or larger is FAT32. [. . .]
Do not spend too much time trying to figure out why this math works. The basis for the computation is complicated; the important point is that this is how Microsoft operating systems do it, and it works. Note, however, that this math does not work perfectly. It will occasionally set a FATSz that is up to 2 sectors too large for FAT16, and occasionally up to 8 sectors too large for FAT32. It will never compute a FATSz value that is too small, however. Because it is OK to have a FATSz that is too large, at the expense of wasting a few sectors, the fact that this computation is surprisingly simple more than makes up for it being off in a safe way in some cases.

Implementacja w Linuksie

Obsługa w jądrze

- Warstwa abstrakcji systemu plików (VFS) pozwala obsługiwać wiele różnych formatów.
- Moduły jądra `fat` i `vfat` — pełna obsługa FAT12, FAT16 i FAT32.

Narzędzia

- **dosfstools:** `mkfs.fat` (alias: `mkfs.vfat`, `mkfs.msdos`, `mkdosfs`), `fsck.fat` (alias: `fsck.vfat`, `fsck.msdos`, `dosfsck`), `fatlabel` (alias: `dosfslabel`).
- **fatcat** (odpowiednik `debugfs`).
- **testdisk** (odpowiednik `debugfs`).
- **fatsort** (sortuje wpisy w katalogach).
- **fusefat** (moduł FAT dla FUSE) i **mtools** (niezależna implementacja FAT w userspace).
- Inne: **fatattr**, **makebootfat**.

Urządzenia dyskowe

Zapis na nośniku fizycznym

- magnetyczny na dysku: CMR, SMR (Shingled Magnetic Recording)
- w komórkach NAND flash: Solid State Drive (SSD)

Kontroler dysku

- odwzorowanie sektorów (sector reallocation, Flash Translation Layer)
- korekcja błędów odczytu (Reed-Solomon, Low Density Parity Check, Turbo Codes, BCH)
- diagnostyka (S.M.A.R.T.)
- obsługa magistrali dysku

Magistrala dysku

- SATA, NVMe, eMMC, SAS, Fiber Channel

Host Controller

- Host Controller Interface (SCSI, ATAPI, AHCI, NVMe, OHCI/UHCI/EHCI/xHCI)

Device Driver

- Tunelowanie np. w USB: SAT (SCSI/ATA Translation), UAS (USB Attached SCSI)

Organizacja dysku

Atomowa jednostka zapisu: sektor

- Sektor logiczny: prawie zawsze 512 B (ale są też np. 520 B)
- LBA (Logical Block Addressing), numerowane od 0
- SATA: LBA ma 48 bitów (128 PB); dawniej 22 bity (2 GB), potem 28 bitów (128 GB)
- interfejs jądra używa 64 bitów (8 ZB); dawniej 32 bity (2 TB)
- Sektor fizyczny: Advanced Format (4 kB), flash page (4–8 kB)
- Większe obszary: SMR (strefy zapisu, zwykle 20–40 MB), SSD (erase blocks, zwykle 128–512 stron, tj. 512 kB – 4 MB)

Wnioski

- System operacyjny przedstawia dysk jako zbiór sektorów numerowanych za pomocą LBA.
- Wyrównanie (alignment) i lokalność przestrzenna ważne dla efektywności zarówno dla dysków magnetycznych, jak i SSD.

Warstwy abstrakcji oferowane przez jądro Linuksa

Partycje

- Najstarsza metoda podziału dysku na „mniejsze dyski”.
- Obecnie popularny GPT. Dawniej MBR, BSD, Sun, SGI.

Partycje logiczne: LVM2

- Można dowolnie „rozcinać”, ale i „sklejać” dyski.
- Jeden z modułów sterownika `dm` (device mapper).

Softraid

- Sterownik `md` (multiple device driver)

Szyfrowanie dysków

- `dm-crypt`
- `dm-integrity`, `dm-verity`

Sterownik dm

Device mapper: ogólny mechanizm transformowania dysków

- Liczne moduły (poza wymienionymi też `cache`, `delay`, `linear`, `striped`, `error`, `zero`, `flakey`, `mirror`, `multipath`, `snaphot`, `zoned` i in.).
- Sporo starego kodu, który istnieje poza `dm`:
  - partycje (`acorn`, `aix`, `amiga`, `atari`, `efi`, `ibm`, `karma`, `ldm`, `mac`, `msdos`, `osf`, `sgi`, `sun`, `sysv68`, `ultrix`)
  - osobny driver `md`

Rozwiązanie we FreeBSD: GEOM

- Struktura obiektowa (moduły — klasy)
- producenci → obiekt GEOM → konsumenci
- klasy implementują wszystkie dostępne transformacje dysków

Utrata danych z dysku

Przyczyny

- Nieatomowość zapisu na dysk
- Uszkodzenie dysku

Rozwiązanie problemu atomowości (metadanych)

- Weryfikacja metadanych (`fsck`)
- Księgowanie (*journalling*)
- *Soft updates*
- COW (*copy on write*)

Rozwiązanie problemu uszkodzeń dysków: redundancja

- hardware RAID
- soft RAID

Uwaga: RAID zapewnia HA, ale nie zastępuje backupów!

Uszkodzenia dysków

Rodzaje nieprawidłowego działania

- całkowita awaria (dysk nie odpowiada)
- *bad sectors* (dysk zgłasza błędy odczytu)
- *silent errors* (*bit rotting*)

Bit rotting

- Teoretycznie niemożliwe, bo dysk używa sum kontrolnych!
- Częsty powód: błędy w oprogramowaniu dysku.

Jak się zabezpieczyć przed „gniciem bitów”?

- RAID nie pomaga
- Sumy kontrolne jedynym sprawdzonym rozwiązaniem
- `ext4` ma sumy kontrolne metadanych; ZFS — wszystkiego
- *scrubbing* — wczesne wykrywanie problemów

Redundant Array of Independent (or Inexpensive) Disks

RAID levels

- **RAID 0** — block level stripes: $n\times$ zapis, $n\times$ odczyt, redundancja: 0, pojemność: 1
- **RAID 1** — mirrors: $1\times$ zapis, $n\times$ odczyt, redundancja: $n-1$, pojemność: $1/n$
- **RAID 5** — block-level stripes with distributed parity: $(n-1)\times$ zapis, $n\times$ odczyt, redundancja: 1, pojemność: $1-1/n$
- **RAID 6** — block-level stripes with double distributed parity: $(n-2)\times$ zapis, $n\times$ odczyt, redundancja: 2, pojemność: $1-2/n$

Stare, niepopularne wersje

- **RAID 2** — bit level stripes with Hamming code: bardzo duże szybkości transmisji, praca synchroniczna (tylko jedna transakcja na raz), kody Hamminga nie mają dużej przewagi nad bitami parzystości
- **RAID 3** — byte level stripes with parity
- **RAID 4** — block level stripes with parity

RAID

Poziomy zagnieżdżone

- **RAID 01** — mirror of stripes
- **RAID 10** — stripes of mirrors
- **RAID 03** — byte-level stripes with parity of stripes
- **RAID 50** — block-level stripes of block-level stripes with distributed parity
- **RAID 60** — block-level stripes of block-level stripes with double distributed parity
- **RAID 100** — stripes of stripes of mirrors

Poziomy niestandardowe

- wiele niestandardowych typów

Zalety i wady hardware RAID

Zalety

- System operacyjny „widzi" macierz jako pojedynczy dysk (nie potrzeba oprogramowania, nie ma kłopotów z bootowaniem itd.).
- Obsługa RAID nie obciążą procesora.
- Podtrzymanie bateryjne — kontroler RAID potrafi dokończyć transakcję mimo utraty zewnętrznego zasilania.

Wady

- Mniej elastyczne, niż rozwiązania software'owe.
- System operacyjny nie jest świadom działania macierzy, nie może planować ułożenia danych na dyskach.

Softraid: sterownik md i interfejs dm-raid

- Zaimplementowany w jądrze Linuksa.
- Nie potrzebuje wsparcia sprzętowego.
- Oferuje RAID 0,1, 4, 5, 6, 10.
- Konfiguracja: `mdadm(8)`.

ext4 — podstawowy system plików Linuksa

Systemy plików często działają latami

- Każdy system plików jest piękny, gdy jest młody.
- Wiele cykli zapisu i kasowania (total writes przekracza wielokrotnie pojemność dysku) — praca alokatora jest bardzo ważna!
- Systemy plików starzeją się w różny sposób (jak wino lub jak mleko).
- `ext4` jest jednym z najlepszych klasycznych systemów plików.

Awarie

- Bardzo dobry `fsck`.
- Bardzo dobre księgowanie (JBD2) — na poziomie bloków.
- Od niedawna sumy kontrolne metadanych i scrubbing.
- Snapshotting — obecnie poprzez LVM2.

RAID a system plików

Wady stosu protokołów md/dm/lvm2/ext4

- RAID nie wie, które sektory są w użyciu, a które nie.
- W razie niespójności RAID nie wie, która wersja jest prawdziwa (nie radzi sobie z gniciem bitów).
- System plików nie wie, na który z fizycznych dysków dane będą zapisane.

Rozwiązanie

- Połączyć system plików z *volume managerem*.

ZFS: The Z File System

- Prace rozpoczęto w 2001 w Sun Microsystems.
- Ogłoszono 14/09/2004, pierwsza wersja 31/10/2005 (Open Solaris).
- Licencja CDDL (open source, ale niekompatybilna z GPL i BSD).
- W 2010 Oracle zamknęło kod.
- Forki open source dalej rozwijane niezależnie.
- Paweł Jakub Dawidek: port do FreeBSD 7 (2008).
- FUSE dla Linuksa (2006), natywny port do Linuksa: ZoL (2008).
- Illumos: fork OpenSolarisa z ZFS.
- Inicjatywa OpenZFS (2013).
- FreeBSD 13 (2021): rebase z Illumosa na OpenZFS. Teraz Linux i FreeBSD mają ten sam upstream.

Założenia projektowe

- „The Enterprise's computer on Star Trek probably runs ZFS" (Michael W. Lucas).
- Liczniki nie 64-bitowe (16 EB, jak w ext4 i Btrfs), ale 128-bitowe. System plików może mieć 288×10^{15} ZB = 288 biliardów ZB = 256 PiZiB.
- Każdy blok (zarówno danych, jak i metadanych) posiada sumy kontrolne.
- Transakcyjność poprzez COW.
- Zalety COW: tanie snapshoty, klony i mirrory, duża lokalność zapisów.
- Wady COW: mała lokalność odczytów.
- Volume manager pozwala na striping, mirroring i stripes with distributed parity (redundancja od 1 do 3 dysków) — odpowiedniki RAID 0, 1, 5 i ich kombinacje.
- Gnicie bitów: problemy w razie uszkodzenia RAM, dlatego zaleca się ECC.
- Wiele niezależnych systemów plików i urządzeń blokowych w jednej przestrzeni dyskowej.
- Kłóci się z hardware RAID.

Trwałe struktury danych

Idea trwałych struktur danych

- Neil Sarnak, Robert E. Tarjan, Planar point location using persistent search trees, CACM 29(7):669–679, July 1986.
- James R. Driscoll, Neil Sarnak, Daniel D. K. Sleator, Robert E. Tarjan, Making data structures persistent. J. Comp. System Sci. 38(1):86-124, February 1989.

Copy on Write

- COW na drzewach jest bardzo efektywny
- Implementacje: ZFS, Btrfs

Virtual devices (vdev)

Rodzaje vdev-ów

- disk
- file
- mirror
- raidz1, raidz2, raidz3
- log (ZIL SLOG)
- cache (Level 2 Adaptive Replacement Cache, L2ARC)

Ponadto:

- spare
- draidz1, draidz2, draidz3; non-default draid (hot spare)
- dedup
- special

Przykład (odpowiednik RAID 10):

```
zpool create mypool mirror da0 da1 mirror da2 da3
```

Działanie ZFS-a

Datasety i zvole

- Dataset — system plików umieszczony na pewnym zpoolu.
- Mountpoint: domyślnie `/zpoolname/datasetname/`.
- Można mieć datasety niemontowalne.
- Zvol — urządzenie blokowe umieszczone na pewnym zpoolu.
- Uwaga: nie konfigurować swap-a jako zvola!

Administrowanie ZFS-em

- ZFS przypomina SystemD: zamiast kompatybilnie dodawać — zastępuje.
- Administrowanie LVM-em zrobione na wzór ZFS.
- Zarządzanie pool-ami: polecenie `zpool`.
- Zarządzanie datasetami: polecenie `zfs`.

Administrowanie ZFS-em we FreeBSD — zpoole

- `geom disk list`
- `camcontrol devlist`; CAM (Common Access Method) — tylko dyski SCSI
- `gpart show`; dokładniej: `gpart list`
- `diskinfo -v <dysk>`
- `zpool create -m none tank da0 da1 da2 da3`
- `zpool list`
- `zpool status tank`
- `zpool destroy tank`
- `zpool create -m none tank mirror da0 da1`
- `zpool add tank mirror da2 da3`
- `zpool online tank da2`
- `zpool scrub tank`

Administrowanie ZFS-em we FreeBSD — datasety, snapshoty i klony

- `zfs create -o mountpoint=/test1 tank/test1`
- `zfs list`
- `zfs create tank/test1/test2`
- `zfs snapshot tank/set1@snap1`
- `zpool set listsnapshots=on tank`
- `zfs clone tank/set1@snap1 tank/set2`
- `zfs set mountpoint=/nowy tank/stary`

- `beadm list`

Administrowanie ZFS-em we FreeBSD — mirrory i szyfrowanie

- `zfs send tank/set1@snap1 > snap1`
- `cat snap1 | zfs receive barrel/set1`

- `zfs create -o encryption=on -o keylocation=prompt -o keyformat=passphrase pool/dataset`
- `zfs get encryption pool/dataset`
- `zfs load-key -r pool/dataset; zfs mount pool/dataset`
- albo: `zfs mount -l pool/dataset`
- `zfs unload-key -r pool/dataset`
- `zfs get keystatus pool/dataset`