Kurs administrowania systemem Linux

Zajęcia nr 14: Systemy plików

Instytut Informatyki Uniwersytetu Wrocławskiego

3 czerwca 2024

Wirtualizacja

Wirtualizacja hardware'u

- *Proces* — wirtualizacja procesora.
- *Plik* — wirtualizacja urządzenia pamięci masowej.

Historia

- CTSS — Compatible Time-Sharing System (MIT, 1961–1973)
- CTSS → ITSS (MIT, 1969), CTSS → Multics (MIT, 1969) → Unix (Bell Labs, 1969)

System plików

Rozwój

- Zbiór danych — plik.
- System plików pozwala umieścić wiele zbiorów danych na jednym nośniku.
- Katalogi (kartoteki, *directories*) — zbiory plików.
- Drzewiasta struktura katalogów.
- Ścieżka dostępu (*path*).

Konstrukcja

- Własności nośnika (sekwencyjny, częściowo sekwencyjny, o dostępie swobodnym).
- Potrzeby użytkownika.
- Metadane.

Narzędzia niskopoziomowe

System plików

- *on-disk format*
- implementacja

Narzędzia do przeglądania/edycji danych binarnych

- `hexdump`, `hd`, `od` plus `less`.
- `dhex`, `hexcompare`, `hexcurse`, `hexedit`, `hexer`, `lfhex`, `hexeditor` (ncurses-hexedit).
- `wxhexeditor` (WX), `ghex` (Gnome), `okteta` (KDE).
- Uwaga na rozmiar plików!

Przykład: tar

- Pamięć taśmowa (taśma magnetyczna $\frac{1}{2}$", długość rzędu 1 km).
- Dawniej np.: 9-ścieżkowa, 1600bpi, 64MB. (Obecnie: zapis serpentynowy lub poprzeczny, pojemności do 15TB.)
- Dostęp sekwencyjny. Zapis: bloki zawierające (domyślnie 20) rekordów po 512B (=10KiB). Przerwy międzyblokowe.
- Metadane pliku: *nagłówek* — jeden lub więcej rekordów umieszczonych przed treścią pliku.
- Brak „spisu treści" lub innych metadanych.
- Metadane: zapisywane tekstowo, napisy zakończone znakiem o kodzie 0, liczby zapisywane ósemkowo. Wypełnienie do granicy rekordu: znak o kodzie 0.

Nagłówek *(us-)tar*

Offset	Rozmiar	Zawartość
0	100	Ścieżka dostępu do pliku
100	8	Prawa dostępu (21 bitów ósemkowo)
108	8	Numer użytkownika
116	8	Numer grupy użytkownia
124	12	Rozmiar pliku w bajtach (ósemkowo)
136	12	Czas ostatniej modyfikacji (epoka Uniksa, ósemkowo)
148	7	Suma kontrolna nagłówka (18 bitów ósemkowo)
156	2	Typ pliku
157	100	Nazwa pliku wskazywanego przez link symboliczny
257	8	Ciąg znaków `ustar\x20\x20\0`
265	32	Nazwa użytkownika
297	32	Nazwa grupy użytkownika
329	8	Numer major urządzenia
337	8	Numer minor urządzenia
345	155	Prefiks nazwy pliku

Dyski

Urządzenia blokowe

- Zapis/odczyt *buforowany* w jądrze, z wykorzystaniem DMA.
- Jednostka adresowania: *sektor*, 512B lub 4KiB (*advanced format*). Adresy: LBA.
- Urządzenia blokowe *fizyczne* i *logiczne*: podział na partycje, LVM, dm, loop.

Systemy plików

- *Plik* — wirtualne urządzenie blokowe.
- Pliki mają *nazwy* i są pogrupowane w drzwiastą strukturę *katalogów*.
- Interfejs jądra: wywołania systemowe `open(2)`, `read(2)`, `write(2)`, `close(2)` i in.
- Interfejs wysokopoziomowy języka C (stdio, glibc): strumienie; `fopen(3)`, `fread(3)`, `fwrite(3)` i in. Dostęp *buforowany* na poziomie plików. Wiele udogodnień, np. `fprintf`, odporność na przerwanie operacji przez sygnały i in.

On-disk format systemu plików

Przechowywanie zawartości pliku na dysku

- Zawartość plików przechowywana w *blokach* (wielokrotność sektora).
- W ext234 rozmiar bloku $= 2^i$ KiB ($i = 1, \ldots, 6$), w niektórych architekturach nie więcej niż rozmiar strony pamięci wirtualnej. W x86: 1, 2 lub 4 KiB. W UFS niekoniecznie, zwykle większe.
- Blok należy w całości do jednego pliku.
  Wyjątek (np. reiserfs): *tail packing*.

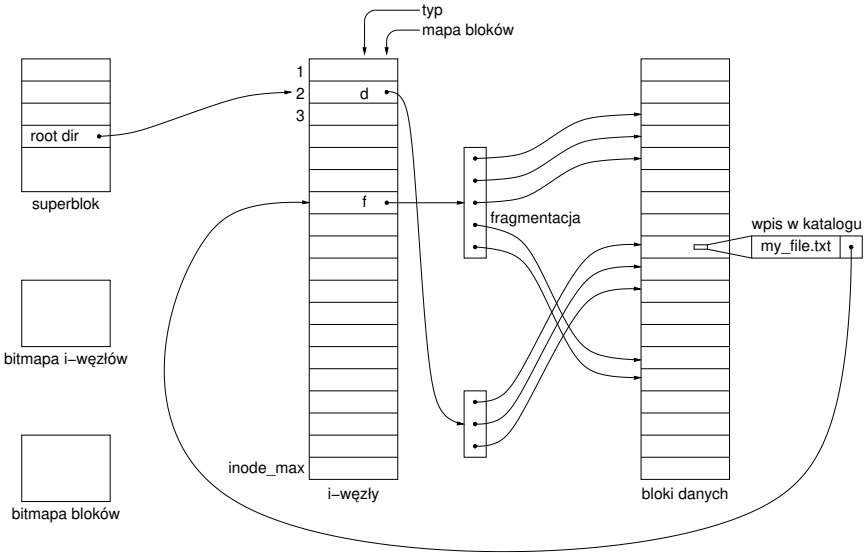
Metadane

- Blok zawierający informacje o całym systemie plików: *superblok*.
- Pliki są numerowane liczbami naturalnymi.
- Baza danych istniejących plików: tablica rekordów (i-węzłów) indeksowana numerami plików.
- I-węzeł zawiera m. in. *mapę bloków pliku*.
- Bitmapa zajętych i-węzłów.
- Bitmapa zajętych bloków.

Katalogi

- Katalog: plik zawierający zbiór nazw plików i przyporządkowanych im numerów plików (i-węzłów).
- Ponieważ katalog jest plikiem, to pojawia się stuktura rekurencyjna: drzewo katalogów.
- Ten sam i-węzeł może występować w wielu katalogach pod różnymi nazwami — dowiązania twarde (*hard links*).
- Potrzeba *link count* (zapisanego w i-węźle). Plik jest usuwany, gdy *link count* spadnie do zera.
- Dowiązania twarde zostały z czasem zabronione dla katalogów (katalogi tworzą *drzewo*).

Struktura systemu plików



Mapa bloków pliku

Wymagania

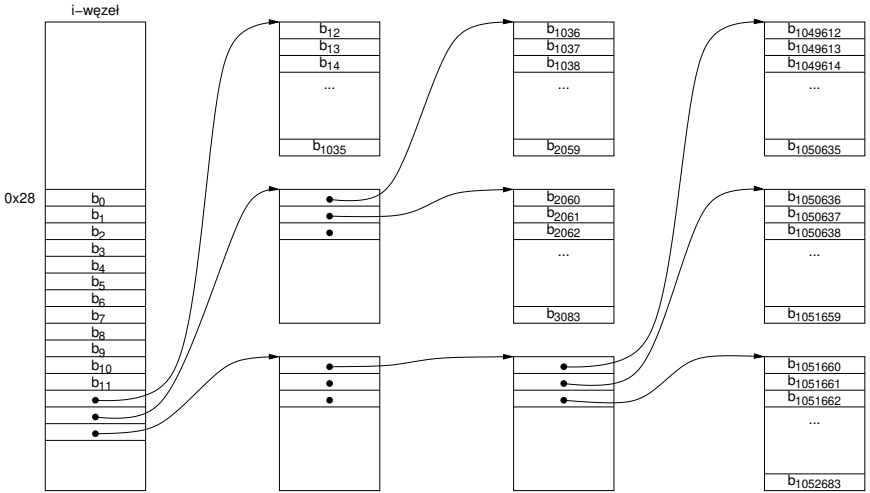
- Rozmiar pliku w bajtach jest zapisany w i-węźle, zatem wiadomo, ile bloków należy do pliku.
- Jednostką adresowania wolumenu jest teraz blok. Adresy bloków są 32-bitowe, chyba że system ext4 ma własność `INCOMPAT_64BIT`.
- Dla bloków 4 KiB i 32-bitowych numerów rozmiar wolumenu $\leq$ 16 TiB.
- *Seek* w pliku: aby przeczytać/zapisać `i`-ty bajt pliku, trzeba zlokalizować `j = i >> (10 + s_log_block_size)` blok tego pliku.
- Wyznaczenie fizycznego numeru bloku na podstawie `j` powinno być szybkie.
- Mapa bloków powinna zajmować niewiele miejsca.
- Adres bloku równy 0 warto zarezerwować na *dziury* (*holes*) w plikach.

Mapa bloków pliku

Klasyczne rozwiązanie (UFS, ext23)

- W i-węźle znajduje się tablica `B` numerów bloków rozmiaru 15.
- `B[0]` ... `B[11]` to numery pierwszych 12 bloków pliku.
- `B[12]` to numer bloku zawierającego tablicę kolejnych $\mathtt{N} = \mathtt{block\_size}/\mathtt{block\_addr\_size}$ numerów bloków pliku (zwykle 1024).
- `B[13]` to numer bloku zawierającego tablicę `N` numerów bloków, z których każdy zawiera tablicę kolejnych `N` numerów bloków pliku.
- `B[14]` to numer bloku zawierającego tablicę `N` numerów bloków, z których każdy zawiera tablicę `N` numerów bloków, z których każdy zawiera tablicę kolejnych `N` numerów bloków pliku.
- Plik może mieć co najwyżej $12 + \mathtt{N} + \mathtt{N}^2 + \mathtt{N}^3$ bloków. Dla bloków 4 KiB i 32-bitowych numerów bloków maksymalny rozmiar pliku to 1074791436 bloków, tj. około 4 TiB.

Mapa bloków pliku



Algorytm wyznaczania numeru bloku

Dane: numer `j` bloku pliku. Szukane: adres bloku `b` na dysku.

```
read_inode(I,n);                  /* wczytaj i-node n do bufora I */
B0 = I + 40;                        /* B0 wskazuje na mapę bloków */
if (j < 12) {
   b = B0[j];                                  /* direct block */
} else if ((j-=12) < N) {
   read_block(B1, B0[12]);
   b = B1[j];                                /* single indirect */
} else if ((j-=N) < N*N) {
   read_block(B1, B0[13]);
   read_block(B2, B1[j/N]);
   b = B2[j%N];                              /* double indirect */
} else {
   j -= N*N;
   read_block(B1, B0[14]);
   read_block(B2, B1[j/(N*N)]);
   j %= N*N;
   read_block(B3, B2[j/N]);
   b = B3[j%N];                              /* triple indirect */
}
```

Mapa bloków

Krytyka klasycznej mapy bloków

- Na każdy blok pliku przypada 32 lub 64 bity mapy bloków (plus adresy pośrednie). Dla bloków 4 KiB i adresów 32-bitowych mapa bloków zużywa ponad 1/1000 rozmiaru pliku.
- Długi czas przeszukiwania mapy (fsck).

Rozwiązanie: zakresy adresów bloków (*extents*)

- *Seek* w pliku musi być szybki!
- Zakresy są stałego rozmiaru, zapisanego w i-węźle (rozwiązanie w FFS, także własność `bigalloc` w ext4).
- Systemowe rozwiązanie w ext4: B+-drzewo zakresów.

Lokalność przestrzenna

Seek time

- Odczyt/zapis sekwencyjny nawet 100 razy szybszy niż przypadkowy.
- Dyski magnetyczne: przesunięcie głowicy + obrót plateru. Seek time rośnie proporcjonalnie do odległości (różnicy LBA).
- Dyski SSD: czas otwarcia bloku kasowania (rzędu 8 MB). Seek time rośnie skokowo po przekroczeniu rozmiaru bloku kasowania.

Rozwiązanie dobre dla dysków magnetycznych

- Kolejno zapisywane bloki powinny mieć bliskie adresy.
- System plików zapisuje na przemian bloki danych i metadanych.
- Wniosek: dane i metadane powinny być blisko siebie.
- Rozwiązanie: grupy cylindrów (UFS), grupy bloków (ext234).
- Bliskie dane względem czasu dostępu (dane i metadane jednego pliku, pliki z tego samego katalogu) umieszczać blisko siebie na dysku — w tej samej grupie bloków.
- Aby to umożliwić bez konieczności późniejszego przesuwania (defragmentacji) — nieskorelowane dane „rozrzucać" po różnych grupach bloków.

Grupy bloków

- Na całym dysku ciągła numeracja bloków począwszy od 0.
- Dysk podzielony na grupy bloków stałego rozmiaru (ostatnia być może niepełna).
- Geometria grup bloków jest opisana w tablicy deskryptorów grup.
- Alokator stara się przydzielić miejsce dla danych i metadanych w tej samej grupie bloków (o ile to możliwe).
- Oryginalnie w UFS: grupy cylindrów: wyrównanie grup bloków do granic cylindrów — minimalizuje konieczność ruchów głowicy.
- Od lat '90 geometria dysków nie jest znana, więc nie ma wyrównania do cylindrów.

Struktura grupy bloków ext234

- Tylko w grupie 0: 1024 bajty nieużywane.
- Superblok rozmiaru 1024 bajty (aktywny w grupie 0, w pozostałych: pasywny *backup*).
- Puste do końca bloku nr 0 (bloki $> 2\,\mathrm{KiB}$).
- Tablica deskryptorów grup.
- Dodatkowe miejsce zarezerwowane na tablicę deskryptorów grup (gdyby powiększono partycję: `e2resize`).
- Bitmapa zajętych bloków tej grupy: 1 blok.
- Bitmapa zajętych i-węzłów tej grupy: 1 blok.
- Tablica i-węzłów; rozmiar:
  `(s_inodes_per_group * s_inode_size) >> (10 + s_log_block_size)` bloków.
- Obszar bloków danych do końca grupy.

Rozmiar grupy bloków i *backupy* metadanych

- Bitmapa bloków zajmuje co najwyżej 1 blok. Stąd wynika maksymalny rozmiar grupy bloków: `(1 << 2 * (10 + s_log_block_size) + 3)`. Dla bloków 4 KiB $= 2^{12}$ B wynosi 2^{27} B $= 128$ MiB.
- Oryginalnie każda grupa zawierała *backup* superbloku i tablicy deskryptorów grup.
- Jeśli system ma własność `sparse_super`, to backupy są tylko w grupie 1 oraz tych, których numer dzieli się przez 3, 5 lub 7.
- Jeśli system ma własność `sparse_super2`, to *backupy* są tylko w grupie 1 i ostatniej.
- Można zrobić backup superbloku i BGDT w pliku na innym dysku.

Struktury danych

Zobacz dokumenty:

- Rémy Card, Theodore Ts'o, Stephen Tweedie, Design and Implementation of the Second Extended Filesystem
- Dave Poirier, The Second Extended File System Internal Layout
- Ext4 wiki: Ext4 Disk Layout
- Kernel wiki: ext4 Data Structures and Algorithms

Narzędzia: e2fsprogs

e2fsprogs

- `mke2fs`, `mklost+found`, `e2fsck`
- `filefrag`, `e2freefrag`, `e4defrag`
- `chattr`, `lsattr`
- `dumpe2fs`, `tune2fs`, `resize2fs`, `debugfs`, `e2image`, `e2undo`, `e2label`, `badblocks`

Inne

- `genext2fs` (systemy wbudowane)
- `fuseext2` (moduł ext2/3/4 dla FUSE)
- `ext3grep`, `extundelete`