

B4B350SY: Operační systémy

Souborové systémy

Michal Sojka¹



2023-12-07

¹michal.sojka@cvut.cz

- 1 Úvod
- 2 Souborové systémy
 - Základy
 - FAT
 - Souborový systém založený na inode
- 3 Žurnálování
- 4 Souborové systémy pro Flash paměti

Co je souborový systém?

- Způsob organizace dat na pevném disku
- Data uložená v pojmenovaných souborech
- Soubory v adresářích (složkách)
- Hierarchická struktura adresářů

Terminologie

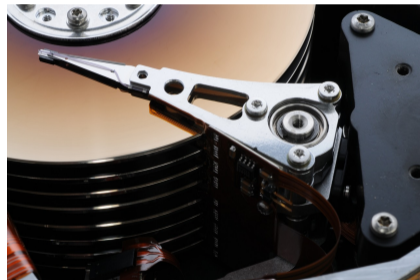
- Data = obsah souborů
- Metadata = pomocné informace ukládané souborovým systémem (bitmapy, inody, superblok, ...)

Požadavky na souborový systém

- Efektivita (nízká režie) – metadata zaberou méně než X % kapacity disku
- Rychlost
- Nízká fragmentace – souvisí s rychlostí (viz dále)
- Spolehlivost (data zůstanou čitelná i po neočekávaném pádu systému) – data mohou být velmi cenná.

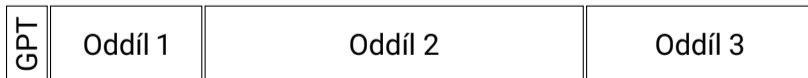
Pevný disk

- Trvalé uložení dat (i bez napájení)
- Rotační (HDD), Flash (SSD)
- Posloupnost bloků (sektorů) určité velikosti
- Každý blok je identifikován číslem



Oddíly (partitions)

- Fyzický disk lze rozdělit na víc logických disků (oddílů, partitions)
- Na začátku disku je tabulka definující typ, (jméno), počáteční a koncový sektor oddílu
 - **Master Boot Record (MBR)** – pozůstatek MS-DOSu, 1. sektor na disku (512 B), obsahuje místo pro 4 oddíly.
 - **GUID Partition Table (GPT)** – modernější, více informací, „neomezený“ počet oddílů.
- Většina souborových systémů využívá **jeden logický disk**
- Ve zbytku přednášky budeme mluvit o logických discích



Otázky

- Jak ukládat adresáře?
- Jak zjistit, ve kterých blocích jsou data daného souboru?
- Jak alokovat bloky na disku při vytváření/zvětšování souborů?
- Jak se vypořádat s chybami a pády systému?
- Jak optimalizovat souborové systémy pro rotační disky a Flash paměti?

Adresáře

- Adresář je seznam dvojic («*jméno souboru*», «*umístění*»)
- Jméno:
 - V UNIXu všechny znaky kromě / a NUL
 - Ve Windows nesmí obsahovat znaky `/\:*"?'<>|`
- Umístění: kde jsou uložena data daného souboru – viz dále
- Třídění seznamu (položek v adresáři):
 - Seznam není uložen setříděný; třídění provádí až program zobrazující adresář uživateli podle jím zadaných kritérií
 - Třídění podle názvu, data přístupu, typu souboru
 - Pomalé otevírání souborů ve velkých adresářích
 - Vyhledávací B-strom
 - Rychlejší

Rozložení dat na disku

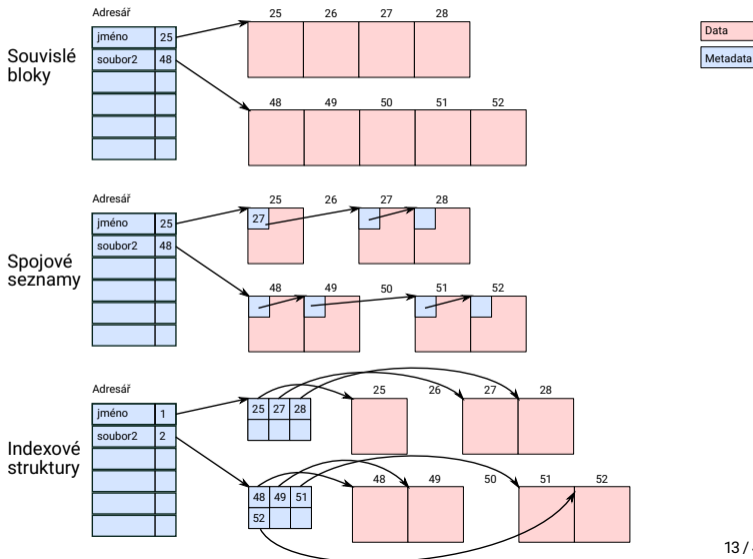
- Souborový systém definuje **velikost bloku** (např. 4 KiB)
 - Prostor na disku je vždy alokován v násobcích velikosti bloku
- **Superblok** určuje umístění kořenového adresáře a další informace o souborovém systému
 - Vždy na předem známém místě (např. 1. blok na disku)
 - Často uložen ve více kopiích
- Informace o **volných blocích**
 - OS musí mít přehled, který blok je volný a který obsazený
 - Podobné jako v alokátorech paměti – např. freelist
 - Typicky bitová mapa (1 bit na blok)
 - Kopie v paměti pro urychlení přístupu (cache)
- Bloky ukládající **obsah souborů**
 - Existuje mnoho způsobů, jak je organizovat

Překlad cesty k souboru

- Co se děje při otevírání souboru „/jedna/dva/tři“?
 - 1 Otevře se kořenový adresář „/“ (vždy se ví, kde se najde – superblok)
 - 2 Najde se v něm záznam „jedna“ a zjistí se jeho *umístění*
 - 3 Otevře se adresář „jedna“ a najde se záznam „dva“ a jeho *umístění*
 - 4 Otevře se adresář „dva“, najde se záznam „tři“ a jeho *umístění*
 - 5 Otevře se soubor „tři“
- Procházení cesty a adresářů po cestě může trvat dlouho
 - Proto je volání `open` odděleno od `read / write`
 - Cesta se prochází jen při `open`
 - Položky adresářů se ukládají do vyrovnávací paměti (dentry cache v Linuxu)

Základní možnosti uložení obsahu souboru

- Obsah souboru je typicky uložen ve více blocích (do jednoho se nemusí vejít)
- Jak se zjistí, které bloky to jsou? Více možností:
 - Souvislé bloky
 - Spojové seznamy
 - Indexové struktury



Základní možnosti uložení obsahu – vlastnosti

- Soubor je vždy uložen v souvislém úseku bloků
 - Podobné alokaci paměti
 - Rychlý přístup k datům (lokalita)
 - Neflexibilní, způsobuje fragmentaci a nutnost přemísťovat soubory
- Spojové seznamy
 - Každý blok obsahuje kromě dat i odkaz na další blok, adresář odkazuje na 1. blok souboru
 - Výhodné pro sekvenční přístup k souborům, nevýhodné pro vše ostatní
 - Nemožnost „mapovat“ data z disku přímo do paměti
 - Jeden špatný sektor na disku (porucha) může způsobit „ztrátu“ zbytku souboru
- Indexové struktury
 - „Indexový blok“ obsahuje ukazatele (čísla bloků) na mnoho jiných bloků
 - Vhodnější pro náhodný přístup, stále poměrně dobré pro sekvenční přístup
 - Může být potřeba použít více indexových bloků

Souborový systém FAT

File Allocation Table

- Starý souborový systém s mnoha nedostatky a omezeními.
- Něco mezi spojovými seznamy a indexovou strukturou.
- Základní jednotka „cluster“ (4–32 KiB)
- Max. počet clusterů: FAT12: 2^{12} , FAT16: 2^{16} , FAT32: 2^{28} , exFAT: $2^{32} - 10$
- Rozložení disku:



- MBR – master boot record (info o soub. systému, tj. velikost, jméno, počet kopií FAT tabulek atd.)
 - FAT1, 2 – dvě kopie FAT tabulky (redundance)
- Specifikace poslední verze FAT (exFAT):
<https://learn.microsoft.com/en-us/windows/win32/fileio/exfat-specification>

Tabulka FAT

Adresář

jméno	0
soubor2	3
soubor3	7
soubor4	34

FAT tabulka

	0	1	2	3	4	5	6	7
8								
16								
24								
32								

Diagram illustrating the FAT table structure. The table is a grid with columns 0-7 and rows 8, 16, 24, 32. Arrows indicate the mapping of files to clusters: 'soubor2' (index 3) points to cluster 3; 'soubor3' (index 7) points to cluster 7; 'soubor4' (index 34) points to cluster 34. Red arrows show a path from cluster 3 to 4 to 5 to 6 to 7. Blue arrows show a path from cluster 7 to 8 to 9 to 10 to 11 to 12 to 13 to 14 to 15 to 16 to 17 to 18 to 19 to 20 to 21 to 22 to 23 to 24 to 25 to 26 to 27 to 28 to 29 to 30 to 31 to 32 to 33 to 34. A green arrow shows a path from cluster 34 to 35 to 36 to 37 to 38 to 39 to 40 to 41 to 42 to 43 to 44 to 45 to 46 to 47 to 48 to 49 to 50 to 51 to 52 to 53 to 54 to 55 to 56 to 57 to 58 to 59 to 60 to 61 to 62 to 63 to 64 to 65 to 66 to 67 to 68 to 69 to 70 to 71 to 72 to 73 to 74 to 75 to 76 to 77 to 78 to 79 to 80 to 81 to 82 to 83 to 84 to 85 to 86 to 87 to 88 to 89 to 90 to 91 to 92 to 93 to 94 to 95 to 96 to 97 to 98 to 99 to 100 to 101 to 102 to 103 to 104 to 105 to 106 to 107 to 108 to 109 to 110 to 111 to 112 to 113 to 114 to 115 to 116 to 117 to 118 to 119 to 120 to 121 to 122 to 123 to 124 to 125 to 126 to 127 to 128 to 129 to 130 to 131 to 132 to 133 to 134 to 135 to 136 to 137 to 138 to 139 to 140 to 141 to 142 to 143 to 144 to 145 to 146 to 147 to 148 to 149 to 150 to 151 to 152 to 153 to 154 to 155 to 156 to 157 to 158 to 159 to 160 to 161 to 162 to 163 to 164 to 165 to 166 to 167 to 168 to 169 to 170 to 171 to 172 to 173 to 174 to 175 to 176 to 177 to 178 to 179 to 180 to 181 to 182 to 183 to 184 to 185 to 186 to 187 to 188 to 189 to 190 to 191 to 192 to 193 to 194 to 195 to 196 to 197 to 198 to 199 to 200 to 201 to 202 to 203 to 204 to 205 to 206 to 207 to 208 to 209 to 210 to 211 to 212 to 213 to 214 to 215 to 216 to 217 to 218 to 219 to 220 to 221 to 222 to 223 to 224 to 225 to 226 to 227 to 228 to 229 to 230 to 231 to 232 to 233 to 234 to 235 to 236 to 237 to 238 to 239 to 240 to 241 to 242 to 243 to 244 to 245 to 246 to 247 to 248 to 249 to 250 to 251 to 252 to 253 to 254 to 255 to 256 to 257 to 258 to 259 to 260 to 261 to 262 to 263 to 264 to 265 to 266 to 267 to 268 to 269 to 270 to 271 to 272 to 273 to 274 to 275 to 276 to 277 to 278 to 279 to 280 to 281 to 282 to 283 to 284 to 285 to 286 to 287 to 288 to 289 to 290 to 291 to 292 to 293 to 294 to 295 to 296 to 297 to 298 to 299 to 300 to 301 to 302 to 303 to 304 to 305 to 306 to 307 to 308 to 309 to 310 to 311 to 312 to 313 to 314 to 315 to 316 to 317 to 318 to 319 to 320 to 321 to 322 to 323 to 324 to 325 to 326 to 327 to 328 to 329 to 330 to 331 to 332 to 333 to 334 to 335 to 336 to 337 to 338 to 339 to 340 to 341 to 342 to 343 to 344 to 345 to 346 to 347 to 348 to 349 to 350 to 351 to 352 to 353 to 354 to 355 to 356 to 357 to 358 to 359 to 360 to 361 to 362 to 363 to 364 to 365 to 366 to 367 to 368 to 369 to 370 to 371 to 372 to 373 to 374 to 375 to 376 to 377 to 378 to 379 to 380 to 381 to 382 to 383 to 384 to 385 to 386 to 387 to 388 to 389 to 390 to 391 to 392 to 393 to 394 to 395 to 396 to 397 to 398 to 399 to 400 to 401 to 402 to 403 to 404 to 405 to 406 to 407 to 408 to 409 to 410 to 411 to 412 to 413 to 414 to 415 to 416 to 417 to 418 to 419 to 420 to 421 to 422 to 423 to 424 to 425 to 426 to 427 to 428 to 429 to 430 to 431 to 432 to 433 to 434 to 435 to 436 to 437 to 438 to 439 to 440 to 441 to 442 to 443 to 444 to 445 to 446 to 447 to 448 to 449 to 450 to 451 to 452 to 453 to 454 to 455 to 456 to 457 to 458 to 459 to 460 to 461 to 462 to 463 to 464 to 465 to 466 to 467 to 468 to 469 to 470 to 471 to 472 to 473 to 474 to 475 to 476 to 477 to 478 to 479 to 480 to 481 to 482 to 483 to 484 to 485 to 486 to 487 to 488 to 489 to 490 to 491 to 492 to 493 to 494 to 495 to 496 to 497 to 498 to 499 to 500 to 501 to 502 to 503 to 504 to 505 to 506 to 507 to 508 to 509 to 510 to 511 to 512 to 513 to 514 to 515 to 516 to 517 to 518 to 519 to 520 to 521 to 522 to 523 to 524 to 525 to 526 to 527 to 528 to 529 to 530 to 531 to 532 to 533 to 534 to 535 to 536 to 537 to 538 to 539 to 540 to 541 to 542 to 543 to 544 to 545 to 546 to 547 to 548 to 549 to 550 to 551 to 552 to 553 to 554 to 555 to 556 to 557 to 558 to 559 to 560 to 561 to 562 to 563 to 564 to 565 to 566 to 567 to 568 to 569 to 570 to 571 to 572 to 573 to 574 to 575 to 576 to 577 to 578 to 579 to 580 to 581 to 582 to 583 to 584 to 585 to 586 to 587 to 588 to 589 to 590 to 591 to 592 to 593 to 594 to 595 to 596 to 597 to 598 to 599 to 600 to 601 to 602 to 603 to 604 to 605 to 606 to 607 to 608 to 609 to 610 to 611 to 612 to 613 to 614 to 615 to 616 to 617 to 618 to 619 to 620 to 621 to 622 to 623 to 624 to 625 to 626 to 627 to 628 to 629 to 630 to 631 to 632 to 633 to 634 to 635 to 636 to 637 to 638 to 639 to 640 to 641 to 642 to 643 to 644 to 645 to 646 to 647 to 648 to 649 to 650 to 651 to 652 to 653 to 654 to 655 to 656 to 657 to 658 to 659 to 660 to 661 to 662 to 663 to 664 to 665 to 666 to 667 to 668 to 669 to 670 to 671 to 672 to 673 to 674 to 675 to 676 to 677 to 678 to 679 to 680 to 681 to 682 to 683 to 684 to 685 to 686 to 687 to 688 to 689 to 690 to 691 to 692 to 693 to 694 to 695 to 696 to 697 to 698 to 699 to 700 to 701 to 702 to 703 to 704 to 705 to 706 to 707 to 708 to 709 to 710 to 711 to 712 to 713 to 714 to 715 to 716 to 717 to 718 to 719 to 720 to 721 to 722 to 723 to 724 to 725 to 726 to 727 to 728 to 729 to 730 to 731 to 732 to 733 to 734 to 735 to 736 to 737 to 738 to 739 to 740 to 741 to 742 to 743 to 744 to 745 to 746 to 747 to 748 to 749 to 750 to 751 to 752 to 753 to 754 to 755 to 756 to 757 to 758 to 759 to 760 to 761 to 762 to 763 to 764 to 765 to 766 to 767 to 768 to 769 to 770 to 771 to 772 to 773 to 774 to 775 to 776 to 777 to 778 to 779 to 780 to 781 to 782 to 783 to 784 to 785 to 786 to 787 to 788 to 789 to 790 to 791 to 792 to 793 to 794 to 795 to 796 to 797 to 798 to 799 to 800 to 801 to 802 to 803 to 804 to 805 to 806 to 807 to 808 to 809 to 810 to 811 to 812 to 813 to 814 to 815 to 816 to 817 to 818 to 819 to 820 to 821 to 822 to 823 to 824 to 825 to 826 to 827 to 828 to 829 to 830 to 831 to 832 to 833 to 834 to 835 to 836 to 837 to 838 to 839 to 840 to 841 to 842 to 843 to 844 to 845 to 846 to 847 to 848 to 849 to 850 to 851 to 852 to 853 to 854 to 855 to 856 to 857 to 858 to 859 to 860 to 861 to 862 to 863 to 864 to 865 to 866 to 867 to 868 to 869 to 870 to 871 to 872 to 873 to 874 to 875 to 876 to 877 to 878 to 879 to 880 to 881 to 882 to 883 to 884 to 885 to 886 to 887 to 888 to 889 to 890 to 891 to 892 to 893 to 894 to 895 to 896 to 897 to 898 to 899 to 900 to 901 to 902 to 903 to 904 to 905 to 906 to 907 to 908 to 909 to 910 to 911 to 912 to 913 to 914 to 915 to 916 to 917 to 918 to 919 to 920 to 921 to 922 to 923 to 924 to 925 to 926 to 927 to 928 to 929 to 930 to 931 to 932 to 933 to 934 to 935 to 936 to 937 to 938 to 939 to 940 to 941 to 942 to 943 to 944 to 945 to 946 to 947 to 948 to 949 to 950 to 951 to 952 to 953 to 954 to 955 to 956 to 957 to 958 to 959 to 960 to 961 to 962 to 963 to 964 to 965 to 966 to 967 to 968 to 969 to 970 to 971 to 972 to 973 to 974 to 975 to 976 to 977 to 978 to 979 to 980 to 981 to 982 to 983 to 984 to 985 to 986 to 987 to 988 to 989 to 990 to 991 to 992 to 993 to 994 to 995 to 996 to 997 to 998 to 999 to 1000

- Jedna položka FAT tabulky má 12/16/32 bitů a odpovídá jednomu clusteru na disku
- Hodnota položky udává číslo následujícího clusteru (konec šipky) nebo -1 značící konec souboru.
- Číslo 1. clusteru se najde v položce adresáře
- Pro urychlení přístupu je tabulka uchovávána v paměti

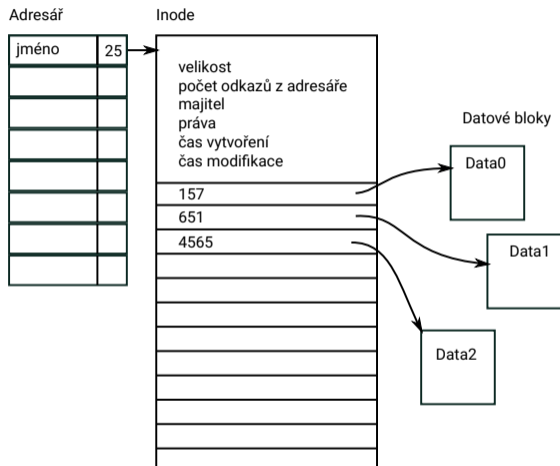
Nevýhody

- Fragmentace
- Omezená velikost (např. $2^{32} \times 4\text{KiB}$)
- Nutnost procházet FAT položky sekvenčně (zpomaluje náhodný přístup u velkých souborů)

Indexový souborový systém

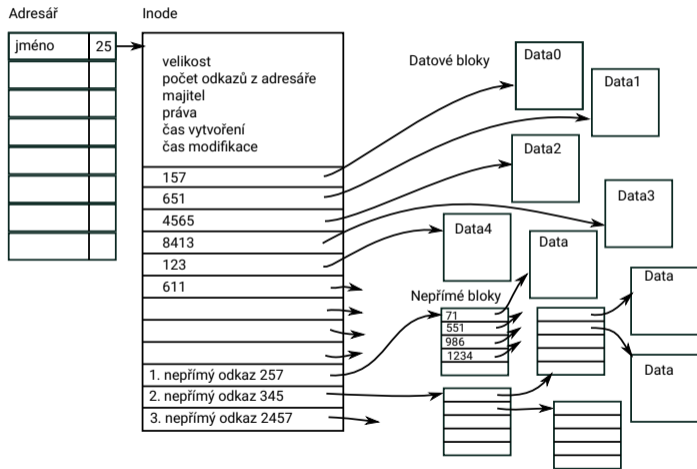
Základ mnoha UNIXových souborových systémů (např. Linuxový ext2 – ext4).

- Metadata o jednotlivých souborech jsou uložena v datové struktuře zvané **inode**.
- Položka adresáře obsahuje kromě jména souboru i číslo (pořadí) inode
- inode obsahuje pevný počet odkazů na datové bloky
 - Z offsetu (pozice v souboru) lze jednoduše spočítat, který odkaz použít pro přístup k datům (dobré pro náhodný přístup)
- Několik inode se vejde do 1 bloku (velikost inode bývá např. 128 B)



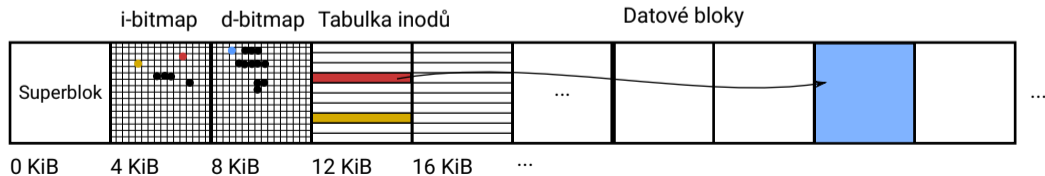
Nepřímé bloky

- Co když je soubor větší, než počet odkazů na datové bloky v inode?
- Odkaz na další bloky nepřímo, přes blok odkazů
- Nepřímé bloky mohou být i v dalších úrovních
 - I u nepřímých bloků lze jednoduše spočítat, který odkaz použít pro přístup k datům (n-ární vyhledávací strom – dobré pro náhodný přístup)



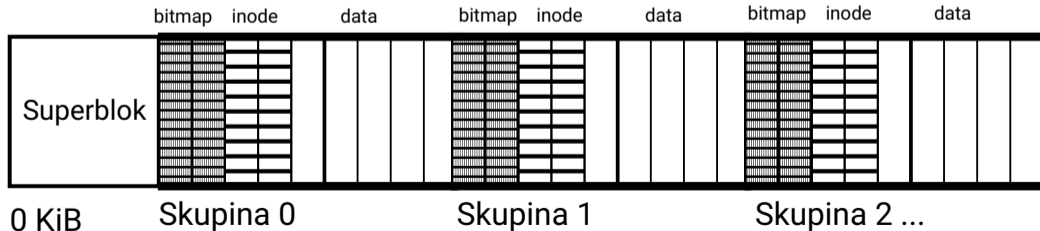
Hledání volného místa

- Jak poznat, který inode je volný (např. při vytváření nového souboru)?
 - Např. sekvenčním procházením všech inode (neefektivní, ale fungovalo by to)
- Jak poznat, který datový blok je volný?
 - Těžko – i blok plný nul může být platným obsahem souboru
- Bitové mapy pro inode a datové bloky
 - každý bit udává obsazenost inodu nebo datového bloku
 - místo prohledávání inodů prohledávám jen bity (rychlejší - např. jedna instrukce = 64 porovnání)

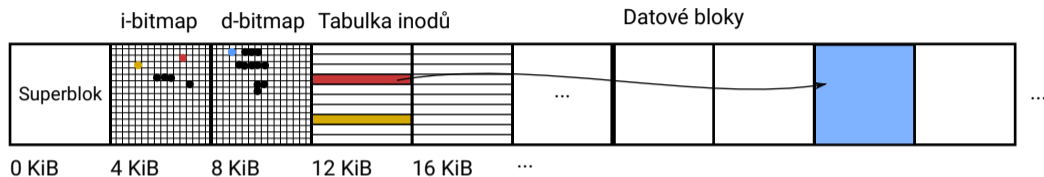


Skupiny (ext2-4)

- Při práci se souborem je potřeba pracovat s bitmapou, inodem a datovými bloky
- Disky (zejména rotační, ale částečně i SSD) přistupují rychleji k blokům uložených blízko sebe
- Co když datové bloky budou až na konci disku?
 - Hlavičky disků musí pořád jezdit mezi začátkem (bitmapy, tab. inode) a koncem disku (data)
- Řešení: skupiny
 - Souborový systém se snaží alokovat datové bloky ve stejné skupině jako inode souboru

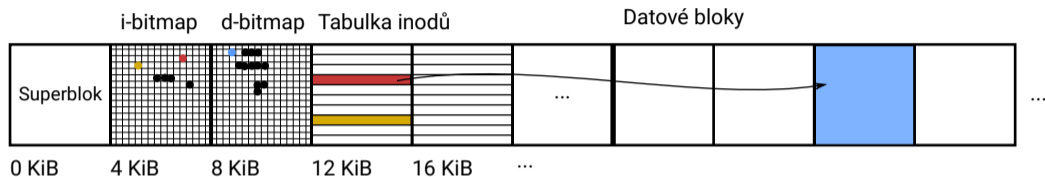


Konzistence dat



- Při zápisu do souboru je potřeba měnit: bitmapy, inode/nepřímé bloky a data
- Hardware disku garantuje atomický zápis pouze jednoho sektoru
- V jakém pořadí bloky zapisovat na disk?
- Co se stane, když dojde k pádu či vypnutí systému v průběhu zapisování při následujících pořadích zápisu?
 - bitmapa, inode/nepřímé bloky, data
 - inode, data, bitmapa
 - bitmapa, data, inode
- Vždy může vzniknout v datech nějaká nekonzistence! (bude porušena integrita souborového systému)
 - Např. můžu zajistit konzistenci při vytváření či zvětšování souborů, ale zkracování

Anketa – prodlužování souboru



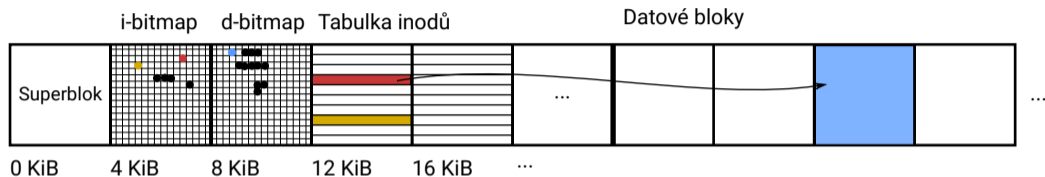
Uvažujme, že souborový systém zapisuje v pořadí:

1) bitmapa, 2) inode, 3) datový blok.

Co se stane, když připojuji data na konec souboru X a dojde k pádu systému mezi 2) a 3)?

- A** Při zápisu dat do jiného souboru může dojít k přepsání části X.
- B** Na konci souboru se objeví „náhodná data“.
- C** Soubor X bude nečitelný.

Anketa 2 – zkracování souboru



Uvažujme, že souborový systém zapisuje v pořadí:

1) bitmapa, 2) inode, 3) datový blok.

Co se stane, když zkracuji soubor X na 0 (truncate) a dojde k pádu systému mezi 1) a 2)?

- A** Při zápisu dat do jiného souboru může dojít k přepsání části X.
- B** Na konci souboru se objeví „náhodná data“.
- C** Soubor X bude nečitelný.

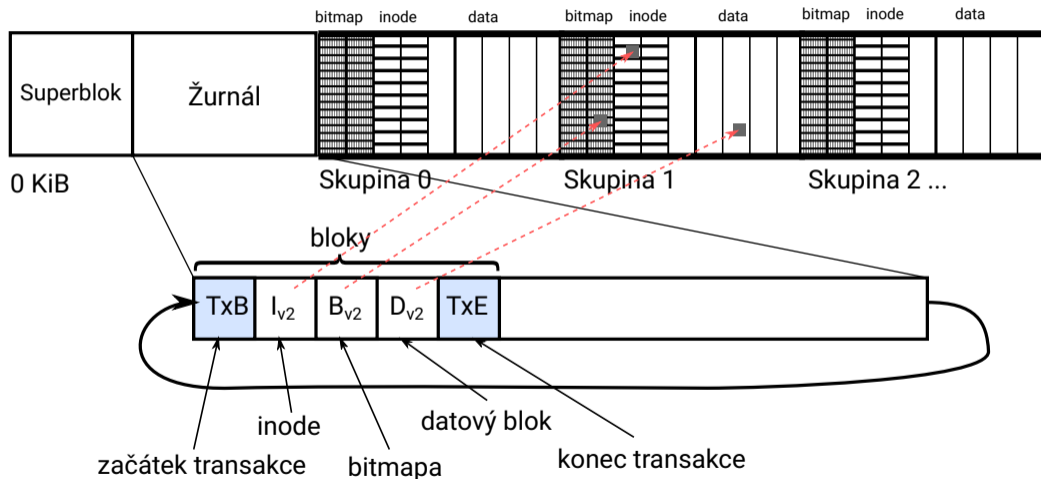
Možná řešení problémů s integritou souborového systému

- 1 **Kontrola souborového systému při startu počítače**
 - projdu všechny inode a nepřímé bloky
 - zjistím, jestli bitmapa volných inode souhlasí se stavem tabulky inode
 - zjistím, jestli bitmapa datových bloků souhlasí s informacemi v inode
 - zjistím, jestli dva inode neodkazují na stejné bloky
 - ...
 - **Pomalé**, zejména na velkých discích!
- 2 **Žurnálování**
- 3 **Souborové systémy používající copy-on-write (btrfs, ...)**

Žurnálovací systém souborů

- Před tím, než se začne souborový systém modifikovat, se uloží seznam potřebných modifikací na vyhrazené místo – **žurnál**
- Pokud dojde k pádu systému, zkontroluje se žurnál, změny disku v něm nalezené se provedou dodatečně
- Žurnálování se někdy nazývá „dopředné logování“
- Implementováno: NTFS, ext3, ...

Struktura žurnálovacího systému souborů (ext3)



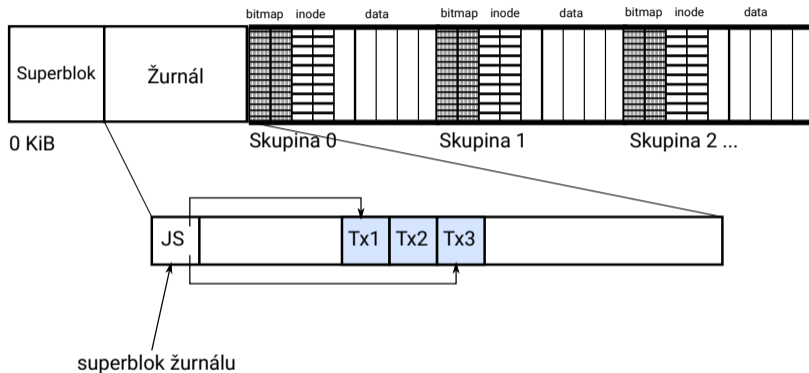
Bezpečný způsob změny souborového systému

- 1 Commit – zapsání transakce do žurnálu
 - TxB : obsahuje id transakce a čísla bloků měněného inode, bitmap a dat
 - I_{v2} : nová verze bloku s inode
 - B_{v2} : nová verze bloku bitmapy
 - D_{v2} : nový datový blok
 - TxE : id transakce, kontrolní součet
- 2 Checkpoint – provedení změn
 - 1 aktualizace bloků v souborovém systému (inode, bitmapy, data)
 - 2 odstranění transakce z žurnálu

Možné scénáře pádu systému a následné obnovy souborového systému

- 1 Do žurnálu se запиše pouze část transakce
 - Souborový systém (SS) je konzistentní a obsahuje původní data
 - Při startu OS se zjistí, že transakce v žurnálu není kompletní (viz další slide) a ignoruje se.
- 2 Do žurnálu zapíšeme celou transakci, ale neaktualizují se bloky SS
 - Při startu OS aktualizujeme bloky SS podle informací v žurnálu
- 3 Zapíšeme celou transakci, aktualizujeme bloky systému, ale neodstraníme transakci ze žurnálu
 - Při startu OS se bloky přepíší ze žurnálu – žádná změna, už zapsané byly a transakce se odstraní ze žurnálu
- 4 Do žurnálu se запиše pouze část transakce (např. TxB , I_{V2} a TxE , bez B_{V2} a D_{V2})
 - **Problém!**
 - HW disků se snaží provádět optimalizace a může změnit pořadí vykonávání příkazů zaslaných OS
 - OS musí disku posílat speciální příkazy (tzv. bariéry), aby se data skutečně zapsala v potřebném pořadí
 - Bariéra garantuje, že příkazy zaslané před bariérou budou vykonány před příkazy zaslanými po bariéře
 - Při zápisu transakce do žurnálu se tedy disku posílá sekvence příkazů:
 TxB , I_{V2} , B_{V2} , D_{V2} , **bariéra**, TxE

Nalezení a aktualizace nevyřízených transakcí



- V jednom okamžiku může vypadat žurnál např. takto (kruhový buffer)
- Superblok obsahuje odkazy na první a poslední platnou transakci v žurnálu
- Commit transakce nebo její smazání se provede atomickým zápisem superbloku se změněnými odkazy do žurnálu

Rychlost žurnálu

■ Pomalé

- Commit: zápis metadat a dat do žurnálu
- Checkpoint: aktualizace inode, bitmapy a dat podle transakce
- Vše se zapisuje na disk dvakrát!

■ Rychlejší:

- Zapsání dat přímo do daného bloku + bariéra
- Commit metadat: Když jsou data uložena, zapsání transakce pro změnu metadat do žurnálu
- Checkpoint: Aktualizace inode a bitmap podle transakce
- Jaké chyby v SS mohou nastat při pádu systému?
 - Data budou částečně aktualizována, ale ne např. velikost souboru

■ Ještě rychlejší:

- Zapsání dat přímo do daného bloku
- Commit metadat: zapsání transakce pro změnu metadat do žurnálu
- Checkpoint: Aktualizace inode a bitmap podle transakce (doufáme, že data zapsána také)
- Jaké chyby v SS mohou nastat při pádu systému?
 - V souboru se mi mohou objevit náhodná data (odjinud)

Všechny módy zajišťují základní integritu SS – nekonzistentní budou maximálně soubory, se kterými se pracovalo při pádu, ne celý SS.

Souborový systém ext4/jbd2

- Uživatel si může zvolit, jaký mód žurnálování se použije
 - **journal**: všechna data i metadata se zapisují skrze žurnál
 - **ordered** (výchozí nastavení): data se zapisují přímo, metadata skrze žurnál po zapsání dat
 - **writeback**: data se zapisují přímo, ale jejich zápis nemusí proběhnout před zápisem metadat (skrze žurnál)
- Typická velikost žurnálu: 128 MiB

Vlastnosti Flash paměti

- Zapisovat lze pouze do vymazaného bloku
- Zapsat na jedno místo lze pouze jednou
- Mazací blok bývá mnohem větší (např. 4 MiB) než blok souborového systému (4 KiB)
- Každý blok garantuje pouze určitý počet přepsání – např. 100 tisíc

Důsledky pro „tradiční“ souborový systém?

- Často se měnící data (např. bitmapy, či FAT tabulka) drasticky snižují životnost paměti
- Změna jednoho bytu v souboru znamená smazání a znovu zapsání 4 MiB
- Garance poskytované žurnálovacím souborovým systémem neplatí pro Flash
 - Commit žurnálu musí vymazat 4 MiB data okolo commitované transakce
 - Pokud systém havaruje mezi smazáním a zápisem, přijdeme o data v žurnálu

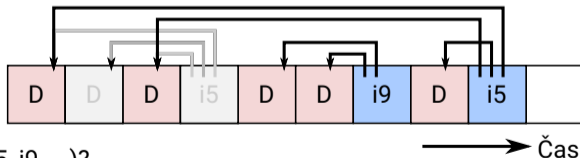
Řešení

- 1 Nepoužívat Flash čipy samostatně, ale v kombinaci s řadičem, implementující „Flash Translation Layer“ (FTL)
 - Mapuje logická čísla sektorů zaslaných OS na bloky flash paměti tak, aby nedocházelo k nežádoucím jevům (např. neustálé přepisování stejného bloku)
 - Implementováno v SSD discích, SD kartách, USB pamětech apod.
 - SD karty/USB paměti mají FTL často optimalizovaný pro souborový systém FAT.
 - Pokud se použije jiný souborový systém, je to pomalé a paměť dlouho nevydrží
- 2 Použít speciální souborové systémy pro Flash paměti
 - Většinou používají princip „copy-on-write“
 - UBIFS, JFFS2, NILFS, ...

Protokolovací souborové systémy

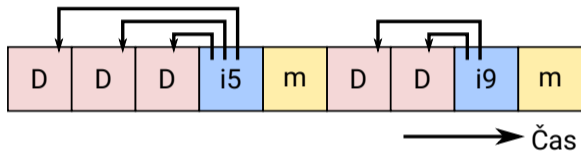
Log-Structured file systems

- Data se čtou převážně z vyrovnávací paměti (page cache)
 - Stačí se zaměřit na operace zápisu – snaha je zapisovat data rovnoměrně po celé oblasti disku
 - Zapisuje se „od začátku do konce“ disku, starší data se nikdy nemění, ale jejich změněné kopie se zapíší na konec.
 - Zápis velkých souvislých bloků dat je velmi efektivní (není třeba znovu zapisovat nezměněná data v mazacím bloku)
 - Stav celého souborového systému je dán zaznamenaným protokolem událostí
- Příklad: zápis dvou souborů na disk a modifikace prvního z nich



- Jak najdeme inody (i5, i9, ...)?
 - Sekvenčně projdeme celý disk a najdeme je. Pomalé :-)

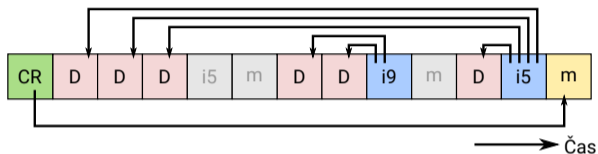
Mapa inodů



- Abychom nemuseli při startu systému procházet celý disk, můžeme si po každé změně SS uložit aktuální „mapu inodů“ (m)
- Mapa inodů obsahuje tabulku pro převod čísel inodů na čísla bloků
- Jak zjistíme, která verze mapy je poslední?

Kontrolní region

Check region (CR)



Čtení souboru

- Přečti kontrolní region
- Najdi pozici mapy inodů (m)
- Najdi inode
- Přečti datové bloky

Zápis souboru

- Zapiš datové bloky
- Zapiš změněnou kopii inode
- Zapiš změněnou kopii mapy inodů
- Aktualizuj kontrolní region

■ CR se pořád přepisuje – nevádí to?

- Nevadí, pokud máme např. SD kartu optimalizovanou pro FAT (viz slide 39), která je optimalizovaná pro častý přepis počátečních bloků s FAT tabulkou
- Využíváno např. F2FS (Samsung)