ZOS CV 05

L. Pešička, 2024

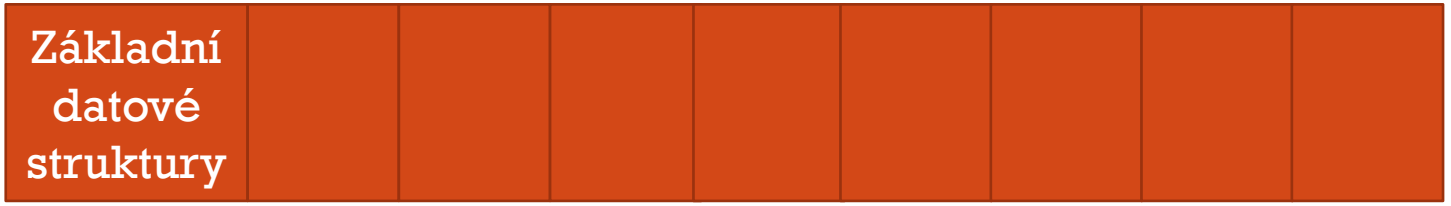
SOUBOROVÉ SYSTÉMY

- Disk se dělí na diskové oblasti (**partitions**)
  disk: /dev/sda, /dev/sdb
  diskové oblasti: /dev/sda1, /dev/sda2, /dev/sdb1
- Každá disková oblast je **naformátována** na určitý filesystém (nebo slouží jako swap) – ext4, xfs, ntfs aj.
- Jak si zobrazím rozdělení disku na oblasti?
  fdisk /dev/sda
- Co je to formátování?
- Jakým příkazem bych formátoval určitou oblast disku?

JAK UKLÁDAT DATA NA DISK?

- Máme diskovou oblast a potřebuje zvolit vhodná pravidla, jak do ní ukládat soubory

- Oblast si rozdělíme na základní datové struktury a dále bloky stejné velikosti, kam budeme ukládat informace o souborech a adresářích:

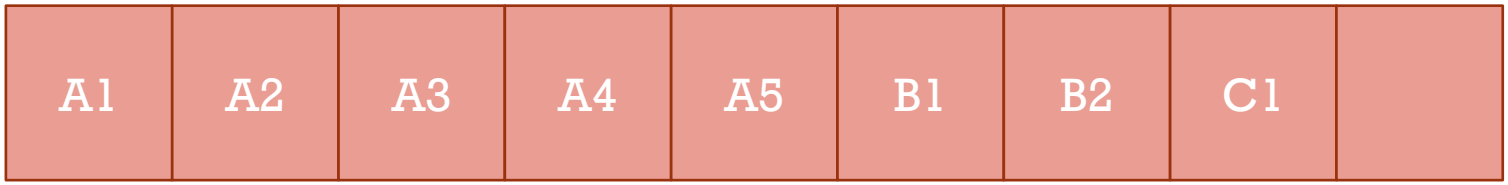


KONTINUÁLNÍ ALOKACE

- Nejjednodušší způsob, jak uložit soubory
- Mějme soubory s těmito velikostmi
  A (4 128 bytů) , B (1 900 bytů) , C (850 bytů).
- Velikost diskového bloku bude pro snadný výpočet 1000 bytů.
- Nakreslete uložení těchto souborů s využitím kontinuální alokace.
- Jaké položky musí obsahovat adresář, abychom mohli k souborům přistupovat?

ŘEŠENÍ

Velikost souboru v bytech potřebujeme, abychom věděli, kolik bytů z posledního bloku je využito



100 105

Adresář musí obsahovat minimálně následující údaje:

Jméno souboru, počáteční blok, velikost souboru v bytech

A, 100, 4128, časové značky, další atributy…
B, 105, 1900, časové značky, další atributy…
C, 107, 850, časové značky, další atributy…

JE VELIKOST BLOKŮ DŮLEŽITÁ?

- Soubor zabere celý blok, i když z něj využije jen část
- Budeme mít soubor velikost 1000B

- Pokud je velikost bloku 512B, ztráta bude 24B
  - Použiju 2 bloky 512B, ztráta bude
    2*512 zabráno – 1000 skutečná velikost = 24
- Pokud je velikost bloku 1024B, ztráta bude 24B
- Pokud je velikost bloku 4096B, ztráta bude 3096B

PROBLÉMY KONTINUÁLNÍ ALOKACE

- Co když bychom chtěli soubor A zvětšit?
  - Problém - není zde místo pro další bloky…
  - Celý soubor A zrušit a zkusit alokovat na konci, zda zde bude místo.
  - Nevhodné pro soubory, které se mohou měnit.

- Lze někde kontinuální alokaci dnes použít?
  - Vypalování CD, DVD, BlueRay

- Má nějaké výhody kontinuální alokace?
  - Snadno najdu požadovaný blok odpočtem od začátku souboru
  - Data nejsou fragmentována

FAT

- Možnost velikosti souboru dynamicky měnit – použití odkazů na další bloky
- Tabulka odkazů na další blok je vyčleněna do samostatné části
- Co obsahují číselné záznamy ve FAT?
  - Číslo bloku, na kterém pokračuje soubor
  - Značka, že se jedná o poslední blok souboru (-1)
  - Značka, že je volné místo (-2)
  - Značka, že odpovídající blok je vadný (-3)

  - Pozn. značky -1, -2, -3 jsou jen pro naše použití.
    U reálné FAT jde o jiné hodnoty.

STRUKTURA DISKOVÉ OBLASTI S FAT

Struktura diskové oblasti:

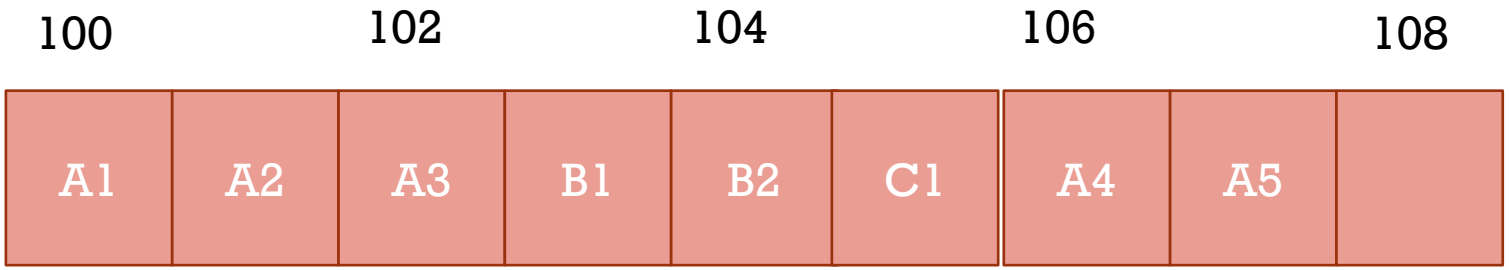
- Boot record a blok parametrů disku
- FAT 1 (velikost je daná počtem datových bloků)
- FAT 2 (kopie FAT 1)
- Datové bloky
  - Na začátku datové oblasti je hlavní adresář (u FAT32 může ležet i jinde)

FAT PŘÍKLAD

- Mějme soubory s těmito velikostmi
  A (4 128 bytů) , B (1 900 bytů) , C (850 bytů).
- Velikost diskového bloku bude pro snadný výpočet 1000 bytů.

- Nakreslete uložení těchto souborů s využitím FAT.
- Jak bude vypadat adresář FAT?

FAT PŘÍKLAD



FAT tabulka:

101	Odpovídá bloku 100
102	
106	Odpovídá bloku 102
104	
-1	Odpovídá bloku 104
-1	
107	Odpovídá bloku 106
-1	
-2	Odpovídá bloku 108

Adresář:

A, 100, 4128
B, 103, 1900
C, 105, 850

UPRAVTE PŘÍKLAD

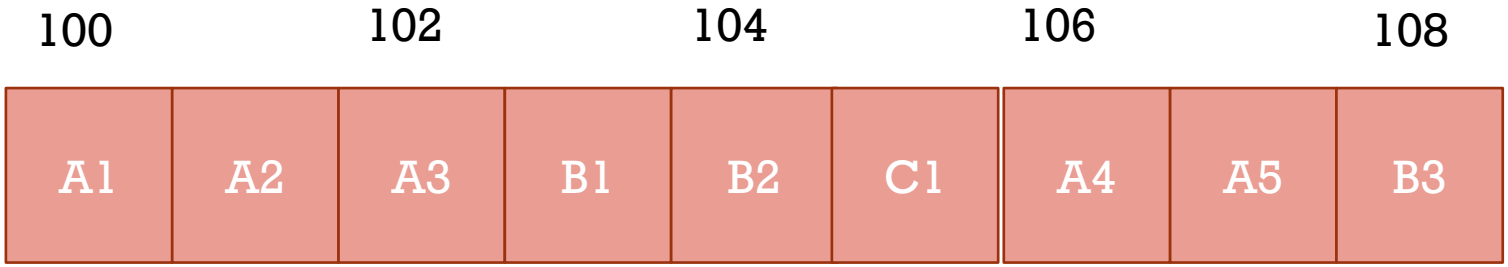
I) Soubor B zvětšíme na 2800 bytů

- Algoritmus projde FAT tabulku a najde volný blok – ve FAT bude označený značkou -2, v našem případě blok 108
- Tento blok využije pro uložení souboru B
- Ve FAT tabulce se změní údaj u bloků 104 a 108
- Změní se i položka adresáře pro B

II) Soubor C smažeme

- Změní se ve FAT tabulce příslušný blok na -2
- Změna se projeví i v adresáři

VÝSLEDEK PO PŘIDÁNÍ SOUBORU B



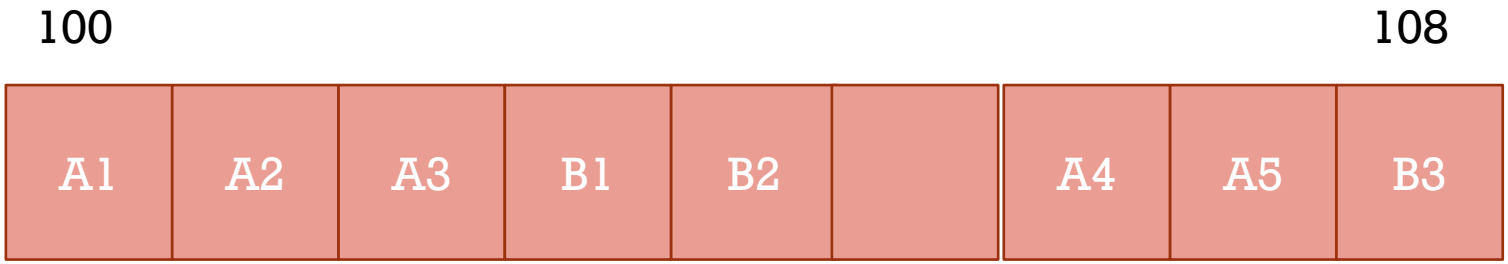
FAT tabulka:

101	Odpovídá bloku 100
102	
106	Odpovídá bloku 102
104	
108	Odpovídá bloku 104
-1	
107	Odpovídá bloku 106
-1	
-1	Odpovídá bloku 108

Adresář:

A, 100, 4128
B, 103, **2800**
C, 105, 850

VÝSLEDEK PO SMAZÁNÍ SOUBORU C



FAT tabulka:

101	Odpovídá bloku 100
102	
106	Odpovídá bloku 102
104	
108	Odpovídá bloku 104
-2	
107	Odpovídá bloku 106
-1	
-1	Odpovídá bloku 108

Adresář:

A, 100, 4128
B, 103, 1900

FAT – značka vadný blok

- Značka vadný blok ve FAT (-3) je upozornění, že odpovídající datový blok je vadný.

- Analogie v realitě – zábradlí, ohraničující výkop
- „Pokud odstraníme zábradlí, výkop nezmizí…“

- Materiály k reálné FATce – podívejte se:
  - http://www.c-jump.com/CIS24/Slides/FAT/FAT.html

FAT - PROBLÉMY

- Velikost souboru neodpovídá délce řetězce ve FAT
  - Řetez je delší nebo kratší
- Řetěze FAT dvou či více souborů se překrývají
  - Používají stejné bloky
  - To je samozřejmě špatně
- Řetěz FAT je zacyklený
  - Např. 3->4, 4->5, 5->4
- Tabulky FAT1 a FAT2 si neodpovídají
  - Oznámit problém
- Limity FAT, např. u FAT32

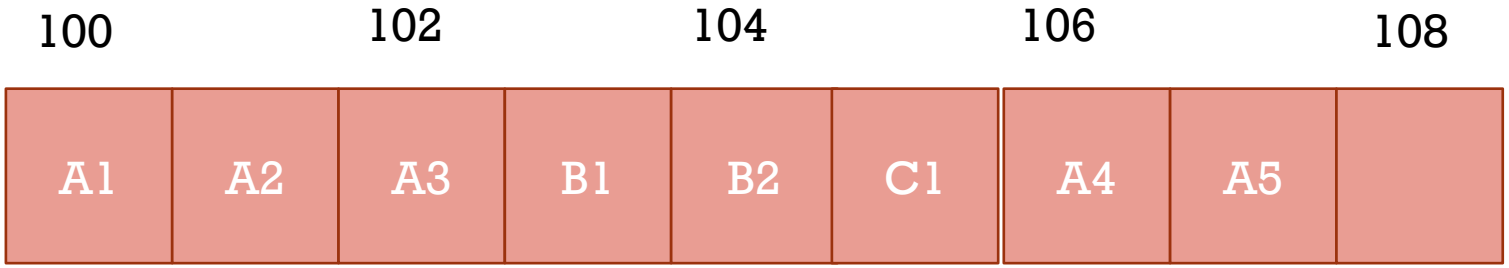
DEFRAGMENTACE

- Týká se jak FATky, tak NTFS
- Datové bloky jednotlivých souborů půjdou za sebou (úplná) nebo bude obsazené místo – volné místo (částečná)
- Předpokládáme defragmentaci na disku, kde je ještě dostatečné místo minimálně na uložení největšího souboru ještě jednou
- Po defragmentaci se nesmí změnit obsah souborů ☺

NTFS - KÓDOVÁNÍ DÉLKOU BĚHU

- Záznamy o souborech jsou uloženy v MFT tabulce.
- Každý soubor zabírá alespoň jeden záznam.
- Umístění datových bloků souboru je popsáno pomocí fragmentů.
- Ideálně jeden soubor představuje jeden fragment.
- Pokud by byl soubor z tolika fragmentů, že by se nevešel do jednoho MFT záznamu, bude pokračující MFT záznam.

NTFS PŘÍKLAD



Soubor A, velikost 4128 bytů
I. fragment (100, 3)
II. Fragment (106,2)

Soubor B, velikost 1900 bytů
I. fragment (103,2)

Soubor C, velikost 850 bytů
I. fragment (105,1)

I-NODE

- Použití i-uzlů pro ukládání souborů je typické pro Unixové filesystémy
- Postupně bylo vícero filesystémů založených na i-uzlech

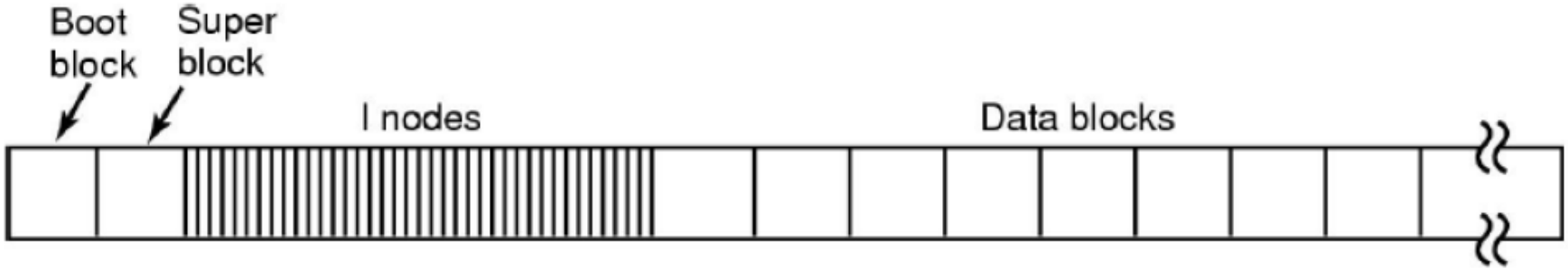
- **1 i-uzel = 1 soubor**
- Jeden i-uzel popisuje právě jeden soubor
- Adresář je také soubor
- V případě hard linků může více jmen z adresáře odkazovat na stejný i-uzel, v i-uzlu je počítadlo odkazů
- Data tvořící obsah souboru jsou popsána jedním i-uzlem

ROZDĚLENÍ DISKOVÉ OBLASTI

- U systému s i-uzly je disková oblast rozdělena na následující části (první verze, později ještě úpravy s blockgroupy):

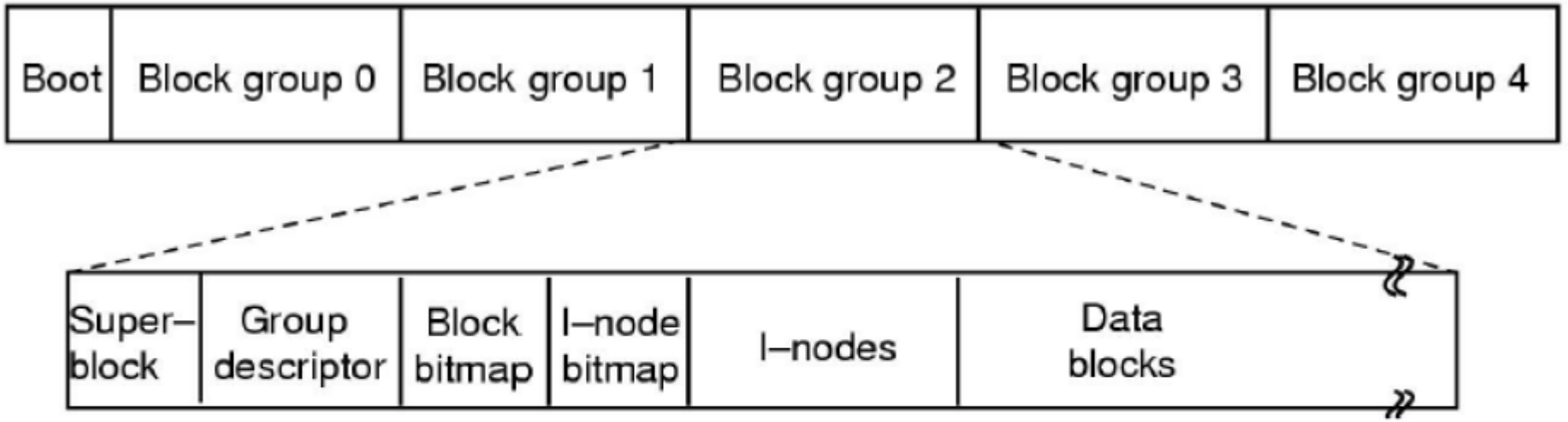
- Bootblock
- Superblock
  - Popis základních struktur filesystému jako je počet i-nodů, velikost alokační jednotky a další
- Oblast i-uzlů
  - Pokud dojdou i-uzly, není možné vytvořit další soubor
- Datové bloky
  - Slouží k ukládání dat a metadat

Původní rozdělení diskové oblasti



Volné bloky – nejprve byl použit seznam volných bloků,
v pozdějších verzích bylo nahrazeno bitmapou signalizující, který blok je volný a který obsazený

NOVÉ ROZDĚLENÍ (EXT2 A DALŠÍ)



skupiny i-nodů a datových bloků v jednotlivých skupinách (block group)
duplikace nejdůležitějších údajů v každé skupině (superblock, group descriptor)

I-UZEL

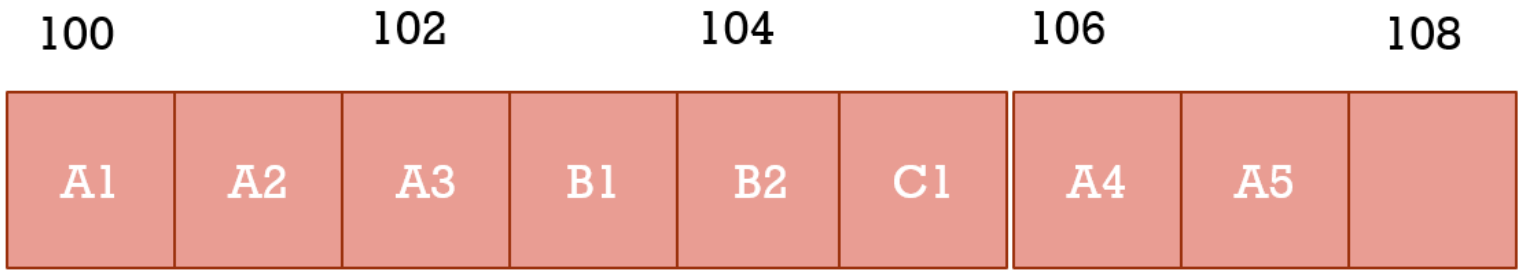
- Pomocí i-uzlu mohu uložit malé i velké soubory.
- Jak je to možné?
- i-uzel obsahuje přímé odkazy na datové bloky souboru
- Přímých odkazů je omezený počet
- Pokud nestačí, jsou nepřímé odkazy 1. , 2. , 3. řádu
- Nepřímý odkaz 1. řádu odkazuje na datový blok, který ale neobsahuje data, nýbrž odkazy na další datové bloky
- Analogicky pro 2. řád (odkaz – odkaz – data) a 3. řád
- Přístup k malým souborům je rychlý (přímo odkaz na datové bloky v i-uzlu)

ADRESÁŘ V SYSTÉMU S I-UZLY

- Adresář obsahuje následující položky
- Název souboru, číslo i-uzlu

- Atributy souboru, jako je
  (velikost souboru,
  přístupová práva, vlastník, skupina,
  časy vytvoření, modifikace)
  jsou už součástí i-uzlu

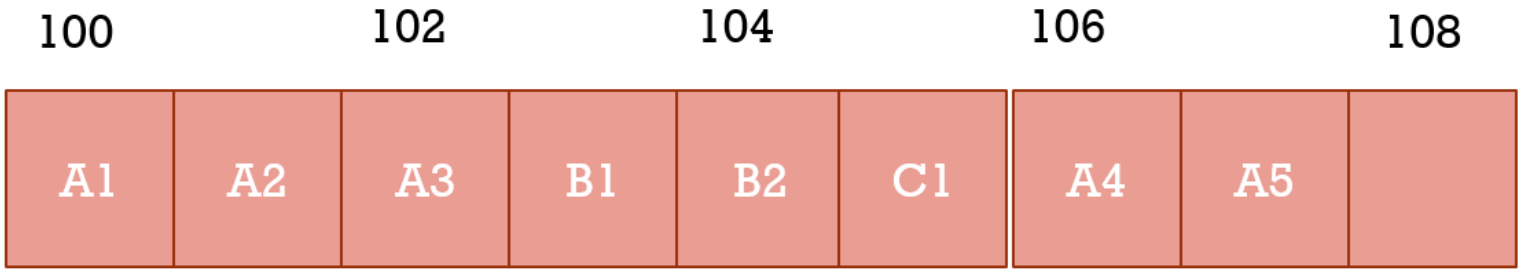
Příklady s i-uzly



- Jaký bude obsah i-uzlů pro soubory A,B,C
  - Pokud přímých odkazů bude 6
  - Pokud přímé odkazy budou jen 3?

- Znázorněte obsah adresáře se soubory A,B,C
- Přidejte ještě soubor D vzniklý: ln B D

I-UZLY - POČET PŘÍMÝCH ODKAZŮ 6



i-uzel 501

atributy
100
101
102
106
107
-
-
-
-

i-uzel 502

atributy
103
104
-
-
-
-
-
-
-

i-uzel 503

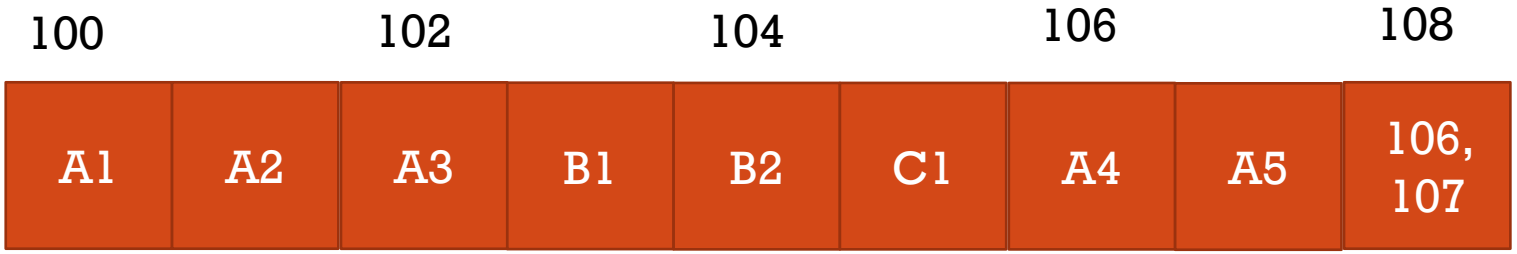
atributy
105
-
-
-
-
-
-
-
-

Atributy: velikost, počet odkazů, přístupová práva, časové značky

Adresář:

A	501
B	502
C	503

I-UZLY - POČET PŘÍMÝCH ODKAZŮ 3



i-uzel 501

Atributy,1
100
101
102
108
-
-

i-uzel 502

Atributy,2
103
104
-
-
-
-

i-uzel 503

Atributy,1
105
-
-
-
-
-

V atributech je explicitně uvedený počet odkazů

Adresář:

A	501
B	502
C	503
D	502

KONTROLA KONZISTENCE

- Kontrolovat lze celou řadu věcí, záleží na filesystému, například:

- Velikost souboru v bytech je v korelaci s počtem odkazovaných datových bloků pro daný soubor.
- U FAT lze dále zkontrolovat, že FAT1 a FAT2 jsou stejné. Pokud ne, dát vědět uživateli, že jsou rozdílné.

- Každý soubor se nachází v nějakém adresáři (není soubor, který by nebyl v žádném adresáři – uživatel by se k němu nedostal). To lze zkontrolovat pro i-uzly.

SEMESTRÁLNÍ PRÁCE

- Souborový systém (disková oblast) bude simulována souborem na disku, např. s názvem myFS.

- ./semPrace myFS
- Při prvním spuštění soubor myFS zatím neexistuje
- Zadáním příkazu format 600MB vytvoří soubor myFS a připraví ho k použití (u běžného příkazu pro formátování se velikost neudává, v naší práci ano, abychom věděli jak velký fs vytvořit)
- Při dalším spuštění již soubor myFS bude obsahovat námi vytvořené soubory a adresáře

FS PO NAFORMÁTOVÁNÍ (FAT)

- Po formátování fs příkazem format vznikne prázdný fs
- Část prostoru zabere jedna či dvě tabulky FAT
  - Ve FAT budou clustery označeny jako volné, s výjimkou prvního, který odpovídá prvnímu bloku a který je hlavním adresářem

- Prázdný fs obsahuje jen položku hlavního adresáře, v které bude jen odkaz na nadřazený adresář (což je sám na sebe)
- Bude využita první položka FAT a první datový blok
- Data hlavního adresáře budou v prvním datovém bloku

FS PO NAFORMÁTOVÁNÍ (I-UZLY)

- Po formátování fs příkazem format vznikne prázdný fs
- Část prostoru zaberou i-uzly, bitmapy
  - Každá položka bitmapy odkazuje na datový blok/i-uzel a říká o něm, zda je volný, obsazený atp.

- Prázdný fs obsahuje jen položku hlavního adresáře, v které bude jen odkaz na nadřazený adresář (což je sám na sebe)
- Bude využit první i-uzel a první datový blok
- Data hlavního adresáře budou od prvního datového bloku

VYTVOŘENÍ NOVÉHO SOUBORU (INCOPY, CP) - FAT

1. Musím najít dostatečný počet volných bloků ve FAT
   - Pokud není, smůla, soubor nelze vytvořit, hlásím chybu
2. Zapíšeme položku příslušného adresáře
   1. Název souboru
   2. Velikost
   3. Počáteční blok souboru (tj. kde začíná řetěz bloků ve FAT odpovídajících našemu souboru)
3. Kopíruji data do příslušných datových bloků a vyplňuji se odkazy ve FAT tabulce

VYTVOŘENÍ NOVÉHO SOUBORU (INCOPY, CP) – I-UZEL

1. Musím najít volný i-uzel
   - Pokud není, smůla, soubor nelze vytvořit, hlásím chybu
2. Zapíšeme položku příslušného adresáře
   1. Název souboru
   2. Číslo i-uzlu, které reprezentuje nový soubor
3. Kopíruji data do příslušných datových bloků a vyplňuji položky i-uzlu (přímé a následně dle potřeby i nepřímé odkazy)
4. Nastavím i příslušné položky v i-uzlu jako je velikost souboru

Přesun – přejmenování souboru (MV)

- Stačí změnit adresářovou položku
  - Jméno nebo přesun do jiného adresáře
- Nemusí se hýbat s datovými bloky patřícími souboru

ZÁKLADNÍ OVĚŘENÍ FUNKCIONALITY

1. Pomocí příkazu incp vložíme do našeho vytvořeného fs nějaký soubor
2. Soubor si vypíšeme uvnitř našeho fs příkazem ls, zkontrolujeme jeho velikost
3. Se souborem následně manipulujeme pomocí kopírování a přesunu
4. Soubor následně outcp vykopírujeme mimo náš filesystém zpátky do operačního systému
5. Porovnáme pomocí cmp či diffu, zda jsou soubory obsahově totožné a nezměnil se žádný bit

ZÁKLADNÍ OVĚŘENÍ FUNKCIONALITY

- Ve stejném adresáři nemůže být soubor i adresář se shodným jménem

- Pokud mám v daném adresáři soubor `a1`, nemůžu zde mít i adresář `a1`

POZNÁMKY

- V celém zadání je poměrně dost volnosti
- Začněte nejprve příkazem formát, vytvořte potřebné datové struktury a hlavní adresář
- Potom implementujte příkaz ls a postupně další

- Není cílem testovat řešení v extrémních podmínkách, ale že systém funguje pro běžné soubory