**PV062 - Organizácia súborov**

# Informačná teória

Informácia – správa, fakt, ktorý o niečom vypovedá, charakterizuje

* Vždy v sebe nesie konkrétnu znalosť ktorá je pre príjemcu nová => znižuje neurčitosť poznania
* Je vždy prenášaná pomocou konkrétneho nosiča (signál, papier, pamäte...) rôznymi podobami (pismena, čísla, symboly, stavy pamäte, ..)
* Aby mohla byť informácia predaná/uchovaná musí byť nejak zakódovaná = prevedená do vhodnej podoby podľa použitého nosiča
* Má zmysel len keď ju vie príjemca interpretovať
* Je vžy o niečom ale nemusí byť precízna = pravdivá či nepravdivá, stále to je informácia
* Nedá sa presne definovať ale je vždy spojená s nejakým fyzikálnym pochodom korý ju nesie, uchováva....

Informačná teória – iniciátorom bol Claude E. Shannon

Správa ktorá nesie informáciu má SYNTAX (formu, je to kvantitatívna stránka informácie, toto študujeme my) a SÉMANTIKU(obsah, kvalitatívna stránka informácie, to neštudujeme)

Intuitívne vieme definovať informáciu ako kvalitu napr ako znalosť odvodenú zo študia, skúseností, zo situácie, faktov...

Z pohľadu softwarového inžinierstva (SE) ju chápeme kvalitatívne.

SE zaujíma hlavne obsah a až potom forma. SE spracováva dáta, informácia reprezentuje význam pridelený údajom ktoré sú v istej správe

My sa ale chceme zaoberať informáciou ako kvantitou. Všebecne rešpektovaná definícia vyjadrujúca mieru informácie je že „informačný objem správy je úmerný miere prekvapenia ktoré správa vyvolá“ . Informácia sa nachádza v správe len vtedy, ak u príjmateľa odstraňuje neurčitosť.

Z pohľadu informačnej teórie chápeme informáciu kvantitatívne. Aby šlo informáciu prenášať, skladovať, ... musí byť merateľná.

Entropia – je miera neurčitosti správy

* Sledujeme výsledok procesu výberu jednej alebo viacej alternatív z nejakej množiny alternatív
* Napr. Zariadenie I produkuje s rovnakou pravdepodobnosťou znaky A, B, C. => zariadenie I vykazuje 3-prvkovú entropiu a keď uvidíme výstupnú hodnotu, neurčitosť mizne a získavame informáciu
* Množstvo získanej informácie odpovedá veľkosti odstránenenej neurčitosti získaním informácie
* veľkosti odstránenenej neurčitosti vieme charakterizovať pravdepodobnosťami
* pr 2. Zariadenie II produkuje s rovnakou pravdepodobnosťou znaky 1 alebo 2 => entropia 2 ... kombinované zariadenie I + II produkuje A1, A2, B1, B2, C1, C2 => entropia 6 ...takto sa ale informácie nemerajú pretože ak prečitam 2 rôzne knihy tak získam množstvo informácií zodpovedajúcich súčtu obsahu kníh a nie kombináciou faktov v nich... miera množstva kombinovaného zdroja teda musí mať aditívny a nie multiplikatívny charakter
* preto je pre mieru množstva informácie najvhodnejšie riešenie => namiesto (3 x 2 ) = 6 sčítame záporné hodnoty logaritmov pravdepodobnosti možných výstupných hodnôt (-log(1/3) + (-log(1/2) = -log(1/6) ... báza logaritmu určuje cieľovú jednotku 2 = bity, 10 = dekadické cifry
* pr 3. Zariadenie III trvalo produkuje jedinú výstupnú hodnotu. Poznaním tejto hodnoty nedôjde k zníženiu neurčitosti ani z toho nezískame žiadnu informáciu keďže vieme čo bude na výstupe nabúdúce => zaiadenie produkuje nulové množstvo inf. (log(1)) = 0
* Množstvo informácie obsiahnutej v správe X súvisí s pravdepodobnosťou jej výskytu
* Požiadavky na mieru množstva informácie :

1. Menej pravdepodobná infrmácia nesie viac informácie, jej získaním sa znižuje väčšia neurčitosť
2. Množstvo informácíí je vždy kladné
3. Množstvo informácií obsiahnuté v skupine nezávislých správ = súčet množstva informácií v jednotlivých správach

Týmto požiadavkam na mieru inf v správe A vyhovujú funkcie :

Neurčitosť, entropia, zdroja správy X, H(X), je rovná množstvu informácie ktoré sa nachádza v správe H(X) = I(X)

Determinované správy majú nulovú neurčitosť; najväčšiu neurčitosť má systém, ktorého správy sú rovnako pravdepodobné.

Entropia závisí na počte stavov systému a na pravdepodobnostiach ich nastania.

SHANNONOVA FORMULA

Ak systém môže mať ***s*** rôznych stavov s pravdepodobnosťami p1, ..., ps (=1) tak entropia diskrétnej náhodnej veličiny X **je**  (tento vzťah má maximálnu hodnotu keď všetky pravdepodobnosti sú rovnaké a teda pi = ½ teda H(X) = log2***s*** [bit]

Vlastnosti miery informácie podľa Shannona :

* Miera je spojitá
* Miera je symetrická
* Entropia má maximum

PR. Zdroj genruje správy zložené zo symbolov A, C, G, T . P(A) = ½, P(C) = ¼ , P(G) = 1/8 , P(T) = 1/8 . Správy sú napr. TAACCAAG, AAAA, ACAACG, ... . Entropia tohto zdroja je H = ½ \* 1 + ¼ \* 2 + 1/8 \* 3 + 1/8 \* 3 = 1.75 (bitov / symbol) (1, 2, 3, 3 (A,C,G,T) sú „prekvapenia“ z výskytu jednotlivých znakov –log2Pi )

FANOVO KÓDOVANIE = minimálne kódovanie symbolov na bity

Kódové slová majú dĺžku úmernú informácii ktoré znaky nesú A = 1, C = 01, G = 000, T = 001

Reťazec 8 symbolov **A**C**A**T**G**A**A**C je kódovaný 14 bitmi => **1**01**1**001**000**1**1**01 teda priemere 14/8 = 1.75 b/symbol

ASCII KÓDOVANIE = uniformne 7b/znak

Rovnaký reťazec ACATGAAC bude kódovaný 56 bitmi (8 x 7) pričom táto správa v priemere nesie 8 \* 1.75 = 14 bitov informácie (56 – 14 = 42 bitov redundancie)

Statické (napr. Textové) informácie sa vyjadrujú diskrétnymi symbolmi => diskrétne správy

Dynamické správy vyjadrujeme v čase spojito formou analógových správ

KÓDOVANIE

Správa je vyjadrená postupnosťou symbolov a tie nadobudajú hodnoty prvkov abecedy

Kódovanie = proces nahradzovania symbolov správy ktoré obsahujú hodnoty zdrojovej abecedy A symbolmi z cieľovej (kódovacej) abecedy AC ....vlastne nahrádzanie symbolov

= je funkcia K : A -> AC+

Napr. Šifrovanie, samoopravné kódovanie, kompresia, ... (strong MB104 vibes)

Abeceda = konečná množina znakov (prvky abecedy, písmená, ..). Pre naše skúmanie platí A ={0, 1}

Slovo = konečná postupnosť znakov istej abecedy. Dĺžku slova |S|, množina všetkých slov nad A A\*, nenulovej dĺžky A+ ....(strong IB102 vibes)

Kódové slovo = prvok AC

Kód C daný kódovaním K je trojica C : (A, AC+, K).

Nutná vlastnosť zmysluplného kódu – je jednoznačne dekódovateľný. Zobrazenie K musí byť prosté = každé dve rôzne zdrojové prvky sú zobrazené na dve rôzne kódové slová. Nutnou! Podmienkou pre dekódovateľnosť kódu je že k prostému zobrazeniu K existuje inverzné zobrazenie.

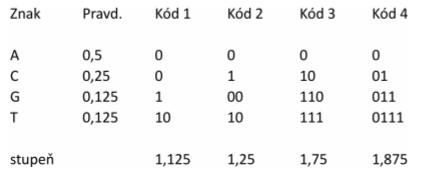
Kód C je jednoznačne dekódovateľný ak všetky reťazce cieľovej abecedy sú jednoznačne dekódovateľné.

KLASIFIKÁCIA KÓDOV

Blokový kód = kód s kódovými slovami pevnej dĺžky napr. ASCII (ASCII je aj jednoznačne dekódovateľny)

Častejšie sa ale používa kód s kódovymi slovami premenlivej dĺžky kvôli redukcii bitov použitých pri obrazení informácie (častejšie používané slová majú kratšie kódové slová...)

Stupeň (rate) kódu C = priemerný počet bitov v kódových slovách kódu C použitých pre kódovanie zdrojových znakov v C

Optimálny kód = má stupeň minimalne prevyšujúci množstvo informácie obsiahnutej v symboloch zdrojovej správy

Neblokový jednoznačne dekódovateľný kód – použijeme už spomínané správy zložené z A, C, G, T. 4 príklady neblokových kódov

Kód 1 – jeho kódovacia funkcia nie je prostá keďže 0 = A ale aj C => ignnorujeme ho

Kódovacie f. Kódov 2,3,4 sú prosté = nesingulárne kódy.

Kód 2 ale nie je jednoznačne dekódovateľný keďže 0000 môže byť AAAA / GG /AAG ...

Kód 3 je prostý, optimálny(má minimálne prevyšujúci stupeň) a naviac **bezprostredne** jednoznačne dekódovateľný – kódové slová sa pri analýze zľava doprava jednoznačne rozpoznávajú, žiadne zo slov nie je prefixom iného slova => tzv. prefixový kód napr UTF-8

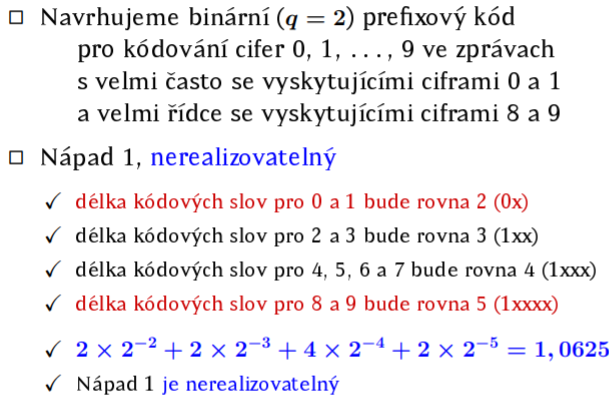
Kód 4 je tiež nesingulárny jednoznačne dekódovateľný nie je ale bezprostredne j.d. – kódové slovo sa rozpozná až po prečítaní úvodnej 0

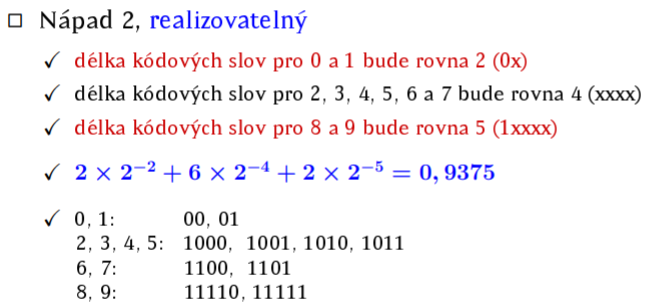
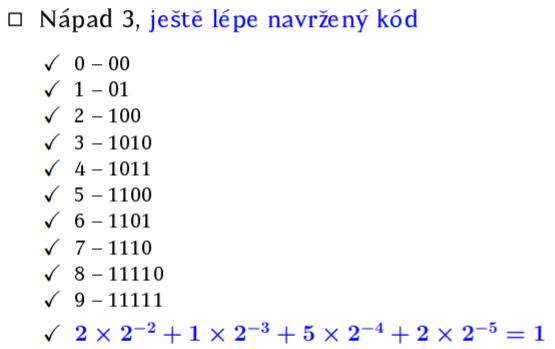
Určenie či ľubovoľný nesingulárny kód je jednoznačne dekódovateľny je algoritmicky neriešiteľný problém ☹

Existujú triedy nesingulárnych kódov ktoré sú dokázateľne jednoznačne dekódovateľné : prefixové, sufixové a blokové kódy

Pre sufixový kód platí že nie je príponou iného kódového slova – je dekódovateľný zprava doľava.

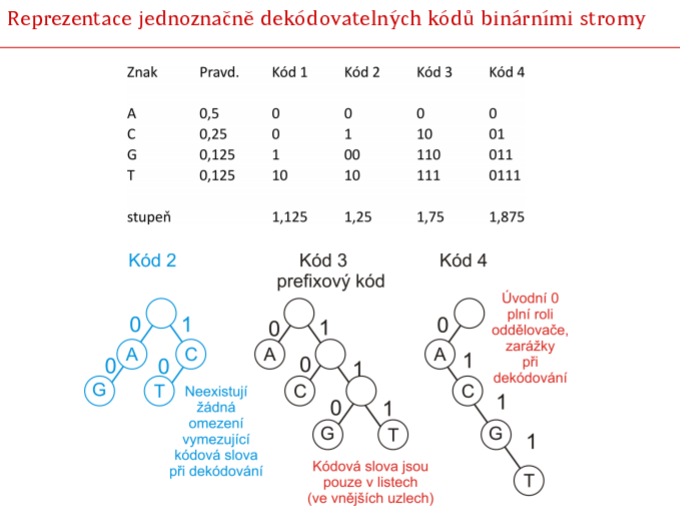
Prefixový q-árny kód s dĺžkami kódových slov d1, d2, ...,dn existuje práve vtedy, keď je splnená Kraftova nerovnosť => kódovacia abeceda obsahuje q znakov a  a platí McMillanova veta : Kraftova nerovnosť platí pre ľubovoľné jednoznačne dekódovateľné kódovanie

KRAFTOVA NEROVNOSŤ, príklad návrhu kódu



**Každý prefixový kód je nesingulárny a jednoznačne dekódovateľný!!**

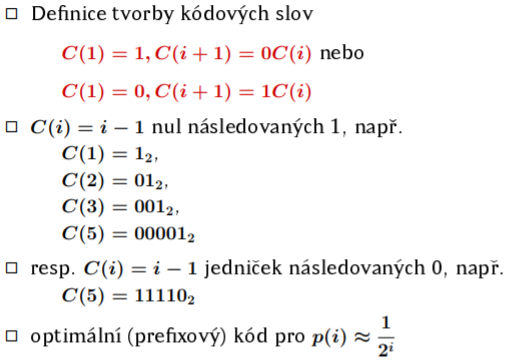
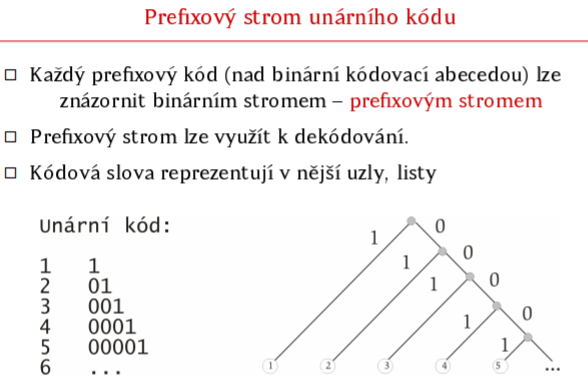
Kedy je kód C optimálny? Keď platí, že preC a náhodnú veličinu X platí kde H je entropia náhodnej veličiny a L je priemerná bitová dĺžka kódového slova



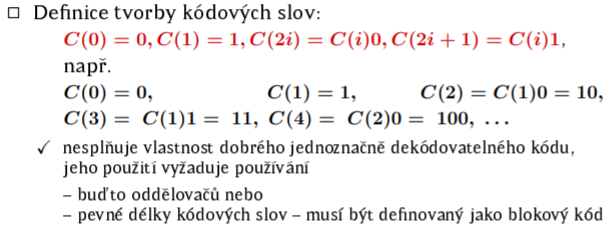
Prečo sa vlastne kóduje? Chceme utajiť obsah, potrebujeme zmeniť veľkosť / typ abecedy, chcme dosiahnuť lepšie využitie kapacity pamäte => kompresia dát = identifikácia a odstránenie redundancie v originálnej správe jej kódovaním do inej abecedy

Cieľom kompresného kódovania je minimalizácia stupňa kódu a zachovanie schopnosti on-line dekódovať kódové slová bez vkladania separátorov

UNÁRNY KÓD

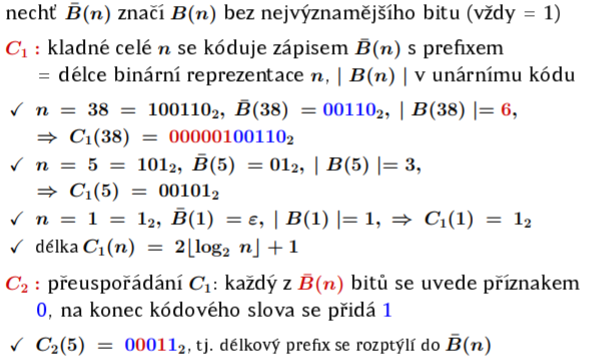
BINÁRNY KÓD

je opimálny pre p(i) = 1/n  
ELIASOV KÓD

je univerzálny kód pre kladné celé čísla.

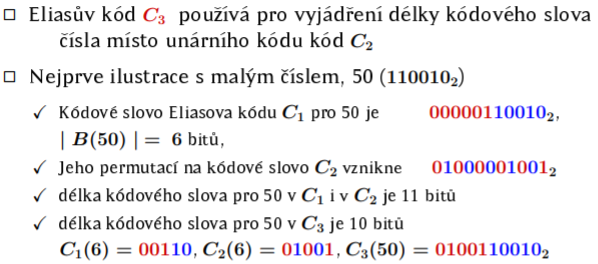
Kódovanie => rozdeľ celé číslo na najvyššiu mocninu dvojky a na zvyšok N. Zakóduj N unárne čo znamená N nul a za nimi 1. Pripoj N dvojkovo zakódované k tejto reprezentácii čísla (38 = 2^5 + 6 ... 6 zakóduj dvojkovo a pred to daj 5 núl a 1 alebo spôsob 2- zakóduj číslo dvojkovo, odčítaj od počtu bitov 1 a toľko núl pridaj pred toto číslo)

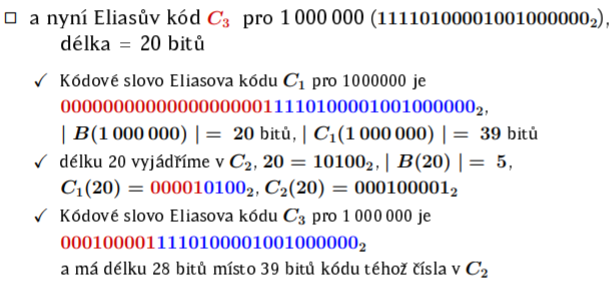
Dekódovanie => čítaj a počítaj 0 kým nenarazíš na 1(N). Táto 1 predstavuje hodnotu 2^N. Teraz už len dočítaj a pripoj ostávajúcich N bitov



Druhý Eliasov kód je preusporiadaním prvého . Zakóduj číslo a medzi každú z číslic vlož 0. Na koniec daj 1.

Kódové slová Eliasovho kódu C1 sú pre veľké čísla dlhé => Eliasov kód C3



priestorovo najvýhodnejší je binárny kód – ten ale nie je jednoznačne dekódovateľný bez použitia separátorov

Unárny kód je prefixový kód => dobrý kód ale len pre malé čísla => pre veľké Eliasov kód

# Kompresia dát

= proces identifikácie a odstraňovania redundancie v dátach

Delí sa na : 1. Bezstratová kompresia = lossless

* + Dekompresiou sa získajú originálne dáta
  + cieľom je zobrazenie do bitového priestoru čo najmenej prevyšujúceho množstvo informácií v originálnych dátach napr. Kompresia textu

2. Stratová kompresia = lossy

- zníženie požiadavku na presnosť rekonštrukcie dát – informácia nevýznamná pre ďaľšie spracovanie sa definitívne odstraňuje napr kompresia obrázkov, zvuku, ..

Kompresia sa vždy týka dvoch algoritmov – kompresného (odstraňuje redundanciu) a dekompresného = rekonštrukčného

Charakteristiky kompresie

* Kódovanie správy dĺžky O(X) [b] na správu dĺžky L(X) [b]
* Cieľom je dosiahnuť L(X) << O(X)
* Ideálne L(X) -> H(X) kde H(X) je entropia
* Menej než H(X) sa správa zakódovať nedá ak nemá byť kompresia stratová
* Kompresný pomer = L(X)/O(X) = 0.75 značí 25% redukciu dĺžky správy
* Kompresný faktor = O(X)/L(X)
* Kompresný zisk = (O(X) – L(X)) / O(X)
* Redundancia správy R(X) = O(X) – H(X)
* Redundancia komprimovanej správy Rk = L(X) – H(X)

Hodnotiace metódy kompresných algoritmov :

* Algoritmická zložitosť
* Pamäťová z.
* Kompresný pomer / faktor / zisk
* Miera stratovosti

Kompresné techniky majú ustálenú dvojfázovú metodiku :

1. Fáza = modelovanie – hľadanie a popis redundancie vo vstupných dátach
2. Fáza = kódovanie – generovanie komprimovaných dat, kóduje sa popis modelu a popis odlišností vstupných dat od modelu do cieľovej abecedy

Klasifikácia modelov – statický model (parametry pre kódovanie sú konštantné) a adaptívny model (parametry sa menia modľa vstupných dat)

Kompresné metody :

1. štatistické – častejšie slovo = kratšie kódové slovo napr Huffman, aritmetické kódovanie, ..
2. slovníkové- udržuje sa zoznam často opakovaných dat napr LZ77, LZ78, LZW

NESTRATOVÁ KOMPRESIA

Rovnaký objem na informácií do menšieho bitového priestoru (napr tie časté a menej časté slová)

STRATOVÁ KOMPRESIA

Efektívnosť kompresie má prednosť pred presnosťou. (kompresia zvuku, obrazu, videa keďže ľudské zmysly nie sú dokonalé, nepostrehnú to)

Aplikácie akceptujúce stratovú kompresiu

-kódovanie reči - GSM (CELP, AMR)

- kódovanie zvuku – MP3, MPEG, AC-3, DAB

- kódovanie obrazu – JPEG, redukcia farieb, RLE, Huffman

- kódovanie videa – DVD, DVB, MPEG

STATICKÁ KOMPRESIA

Nemenný model, nezávisí na aktuálne komprimovaných datach

ADAPTIVNÁ KOMPRESIA

Dynamicky sa mení, rešpektuje vlastnsti aktuálne komprimovaných dat

KROKY KOMPRESIE

Model = spôsob vyjadrenia správy pre účely generovania a rozpoznávania správ

* forma vytvorenia modelu : staticke metódy kompresie alebo adaptívne
* možná funkčná klasifikácia modelov :
  + - model konečného kontextu
    - model konečných stavov – neštuduje sa
    - model s gramatikou – neštuduje sa

ŠTATISTICKÉ MODELY KONEČNÉHO KONTEXTU

* model nultého stupňa – pravdepodobnosti výskytov všetkých znakov sú rovnaké preto sa tento model nepoužíva
* model prvého stupňa – pravdepodobnosti výskytov znakov sú rôzne => najširšie uplatnenie
* model druhého stupňa – model zohľadňuje oravdepodobnosti výskytov dvojic znakov
* model n-tého stupňa – model pracuje s pravdepodobnosťami n-tic znakov

KLASIFIKÁCIA KOMPRESNÝCH METÓD

* základné techniky – intuitívne metódy, Brailovo písmo, RLE, Baudotov kód
* štatistické metódy – Shannon-Fanovo, Huffman, aritmetické
* slovníkové metódy – LZ77, LZ78, LZW

BRAILOVO písmo – 1820, 6bodov v obdĺžníku tj. 6bitov, tj. 2^6 = 64 kódových slov. Písmen je iba 26 preto sú ostatné použité na zakódovanie frekventovane používaných slov

BAUDOTOV kód – 5bitový kód

AD HOC textová kompresia – vynechanie medzier + bitová mapa polohy medzier, ...

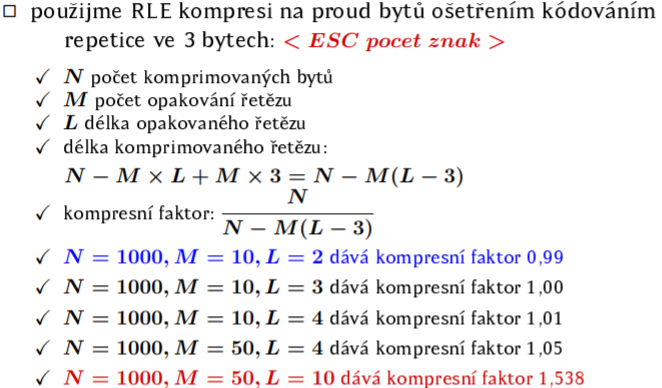
SLOVNÍKOVÁ predná kompresia

MACWRITE kódovanie (word processor) – symbolom je slabika obsahujúca buď jeden ASCII znak alebo 2 komprimované znaky -> to sú 4bitové kódy pre 15 najčastejšie používaných znakov; pre zobrazenie ostatných znakov sa používa znak ESCAPE nasledovaný v ďaľšej slabike plným ASCII znakom

RLE kódovanie (Run Length Encoding) – nahrádza opakujúce sa susedné symboly počtom opakovaní(jeden bajt) a špecifikáciou opakovaného symbolu (druhý bajt) => „bytový prúd“

* + používaná historicky v modemoch MNP class 5
    - * Dosahoval až 200% efektívnosť
      * Princíp : 3 identické byty vyslané bezprostredne za sebou spustia RLE. Za tieto 3 byty sa vsúva repetičný indikátor. Ak sa v opakovaní nepokračuje indikátor indikuje 0 AAA -> AAA0 (expanzia správy) . Ak sa opakujú 4 byty zachováva sa pôvodná dĺžka správy AAAA -> AAA1. Ku kopmresi dôjde ak sa opakuje >=5 bytov AAAAA -> AAA2
      * Vylepšenie : in flight coding – pre kompresiu správy sa používa štatistika výskytu znakov (opakovane sa vyskytujúce znaky dostanú kratšie kódové slovo); ide o adaptívny model
      * Kompresia neopakovaných znakov textu je kontraproduktívna !!
  + ITU-T T4(Group 3)
    - * Štandard faxov
      * Princíp : stránka je zostavená z bielych a čiernych pixelov,predpokladá sa, že každý riaodk začína bielym pixelom a každý riadok je tvorený striedaním postupností b&w pixelov => to umožňuje prenášať iba dĺžky týchto postupností bbbbwwbbbbb = 1,4,2,5

Efektívnosť metod RLE



BINHEX 4.0 FORMAT – 2 krokové kódovanie :

1. RLE : súbor je prúd bytov kde môže nastať 3-255 repetícií (n) bytu XX16 sa kóduje trojicou <XX169016n>, n=0 indikuje žiadnu repetíciu

00 11 22 33 44 55 66 77 -> 00 11 22 33 44 55 66 77

11 22 22 22 22 22 22 33 -> 11 22 90 06 33

11 22 90 33 44 -> 11 22 90 00 33 44

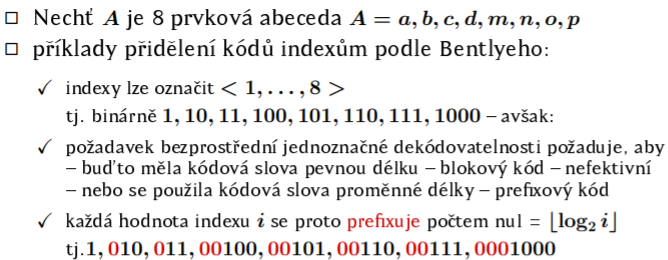
11 22 90 00 33 44 -> 11 22 90 00 00 33 44

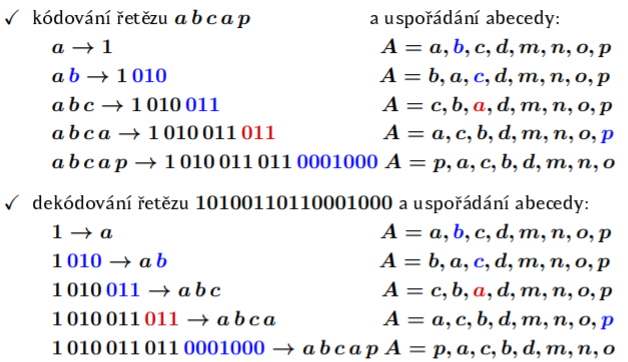
1. Kódovanie do textovej formy – na 7bitové ASCII znaky; RLE súbor je prúd bitov ktorý sa člení na 6bitové prvky

64 bitový prvok = ukazateľ do BinHex tabuľky

BENTLEYHO kódovanie – tzv Move-to-Front Coding – dynamicky sa vytvára a udržuje abeceda A ktorej prvky sa v komprimovanom texte vyskytujú najčastejšie a práve kódovaný prvok sa presúva na začiatok abecedy

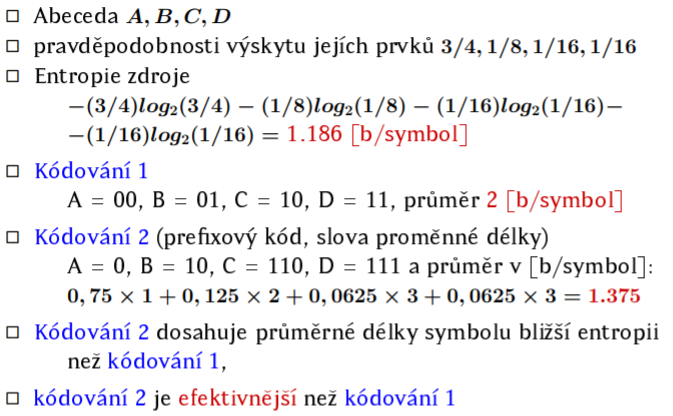
* + uplatňuje sa lokálna adaptívna – efektívna v prípadoch keď sa na jednom mieste koncentrujú identické symboly
  + ako prideliť menším indexom kratšie kódy než väčším indexom?





ŠTATISTICKÉ METÓDY

* Kompresný teorém = ak je H [b/znak] entropia zdroja geenrujúceho správy, tak je možné nestrátovo komprimovať zdrojový text až do zakódovania každého znaku zdrojového textu v priemere H bitmi pričom platí, že H môže byť << log2N kde N=počet prvokv výstupnej abecedy
* Princíp :



* + Shannon- Fanovo kódovanie
    - * kód je reprezentovaný bin.stromom vybudovaným nad znakmi vstupnej abecedy zoradenými do postupnosti podľa klesajúcej pravdepodobnosti výskytu – tento strom vznikne rekurentným delenim postupnosti na 2 časti (podľa približne rovnakej pravdepodobnosti výskytu) zhora dole pričom jedna postupnosť bude vždy začínať 0 a druhá 1
  + Huffmanovo kódovanie
    - * Vytvára kódovací strom opačným smerom než Shannon-Fanova metóda teda zdola nahor => Huffmanov strom
      * Má typické vlastnosti optimálnych prefixových kódov (vyššia pravdepodobnosť => kratšie slová)
      * Prvkom abecedy X sa priraďujú postupnosti prvkov abecedy Y napr X = 0, ..., 255; Y = 0,1
      * Postupne sa zlučujú prvky s najnižšou pravdepodobnosťou výskytu do nového prvku atď smerom hore, Koreň stromu reprezentuje všetky prvky vstupnej abecedy a má p výskytu 1
      * Algoritmus na obrázkoch <https://www.fi.muni.cz/usr/staudek/vyuka/filesys/02_komprese_dat.pdf> 63 – 70
      * Adaptívna varianta = algoritmus FGK – vytvára sa bin.strom s ohodnotenými vrcholmi ktorý musí ať súrodenecký vlastnosť = každý vrchol okrem koreňa ma súrodenca a vrcholy vieme usporiadať do neklesajucej postupnosti podľa ich váh tak, že súrodenci sú vždy na susedných miestach

Ku každému uzlu doplníme váhu a číslo uzlu – váha listu je počet doterajších výskytov znaku, váha vnútorného uzlu je súčet vah potomkov; číslo uzlu je y intervalu <1, 2,..., 2n-1>

Algoritmus na obrázkoch – rovnaký link 74-75

* + - * Ako kódovať znaky ktoré sú načítane prvykrát?
        + 1. Pri počiaočnej inicializácii sú do Huf.stromu vložené znaky s výskytom 1
        + 2. Pri počiatočnej inicializácii je vložený špeciálny znak ESC nasleedovaný daným znakom, potom je vložený nový list odpovedajúci znaku
      * Korekcia stromu, kodér algoritmus rovnaký link 77-94 (nie je to príliš pekné, komu sa nechce nech nepozerá)
      * Vlastnosti Huff.kódovania : kód má minimálnu priemernú dĺžku kódoveho slova, pre istú triedu textov vieme vytvoriť kódovú knihu, efektívnejsšie než Shannon-Fan
  + aritmetické kódovanie
    - * celá správa je komprimovaná do jedného kódového slova z intervalu <0, 1)
      * je vhodné pre prípady keď sa v distribučnej funkcii objavujú veľké pravdepodobnosti
      * Princíp : interval 0-1 sa rozdelí na N intervalov kde N=počet prvkov v zdrojovej abecede. Šírky intervalov sú úmerné pravdepodobnostiam výskytu prvku(cdf). Správa začínajúca i=tým prvkom vstupnej abecedy je reprezentovaná číslom z intervalu vymedzenom i-tému prvku vstupnej abecedy
      * Je potreba zakódovať informáciu o ukončení pretože dekodér musí vedieť kedy má skončiť s delením intervalu
      * Nejaké príklady, upletnenie algoritmu na AK Rovnaký link 97-111
      * Dekodér môže použiť aj predom pripravený pravdepodobnostnź model napr na kódovanie kníh v češtine => adaptívna verzia AK
  + slovníkové metódy
    - * V komprimovanom texte si vyberajú reťazce znakov a každý reťazec sa kóduje nejkým príznakom pomocou slovníku
      * LZ77
        + WinZip, PNG, ...
        + slovník tvorí istá spracovávaná časť textu = posuvné okno – má 2 časti : už zakódovanú časť a prehliadané okno
        + výstupom je trojica : ukazateľ počiatku vzoru v slovníku, dĺžka vzoru, kódové slovo ďaľšieho symbolu za kódovaným textom
        + 123 - 127
      * LZ78
        + Slovník sa udržuje formou vyhľadávacieho stromu (uzly = reprezentácie slov v slovníku), postupnosť ohodnotených hrán = kódové slová znakov tvoriacich fráze, indexy frází = číslované uzly
        + výstupom je dvojica : index vzoru v slovníku, kódové slovo budúceho symbolu za kódovaným textom
        + 133-149
      * LZW
        + slovník sa buduje ako rozšírenie kódovacej abecedy
        + GIF, TIFF, PDF
        + výstupom je jediný údaj : index vzoru v slovníku
        + 151-171

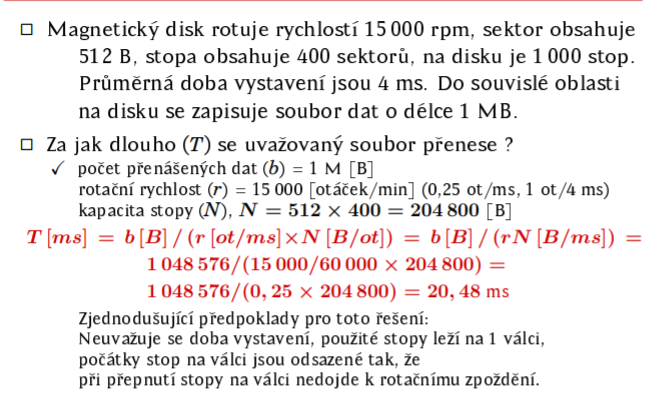
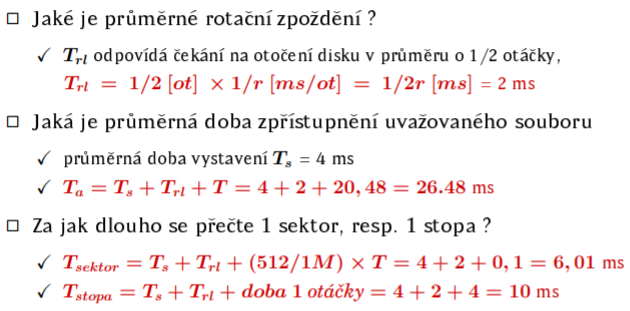
# Pamäte

VNÚTORNÉ PAMATE

* CACHE – najrýchlejší a najnákladnejší spôsob ukladania
  + energetická závislosť
  + spravuje HW
* HLAVNÁ PAMAŤ – pamäť dostupná procesoru pre uloženie programu a dát
  + rýchly prístup
  + buď je príliš malá alebo príliš drahá
  + obsah pamäte sa stratí ak dôjde k zlyhaniu systému

VONKAJŠIE PAMATE

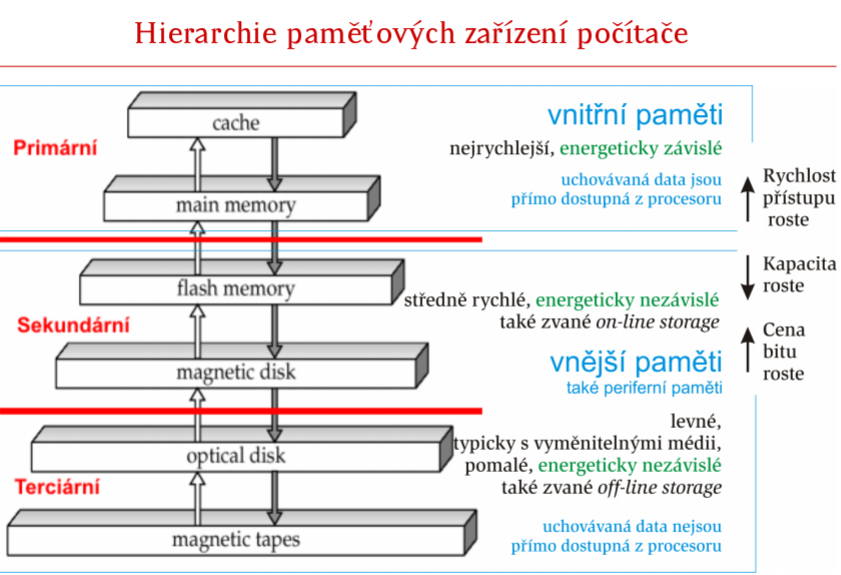
Dáta musia byť pre spracovanie presunuté do hlavnej pamäte a pre uloženie zapísané späť !!!!

* FLASH PAMAŤ
  + data prežijú pri výpadku pamäte
  + dáta sa dajú zapísať je jedno miesto iba raz, ale miesto sa dá vymazať a zapísať do do neho dáta znovu
  + zápisy sú pomalé, čítanie je zhruba rovnako rychlé ako z hlavnej pamäte, mazanie je pomalšie
  + digitálne foťáky, telefony, USB klávesnice
* MAGNETICKÝ DISK
  + dáta sú uložené na rotujúcom disku a čítané/zapisované magneticky
  + sekundárna pamäť, nízka cena a veľká kapacita
  + používa sa hlavne na dlhodobé ukladanie dát
  + Diskovú zostavu tvorí niekoľko magnetických diskov (1-5) nasadených na os
    - Čítacia / zápisová hlava
    - Na jeden povrch jedna čítacia/zápisová hlava, všetky hlavy sú na spoločnom vystavovacom ramene
    - Nachádza sa vo veľmi blízkej vzdialenosti od povrchu
    - Rotáciou diskov vzniká na povrchu záznamová stopa
      * Stopa sa delí na sektory
        + Najmenšia jednotka dát, ktorá sa dá zapísať/čítať
        + delenie na sektory je fixne kódované do diskového povrchu a nedá sa meniť
        + Typicky 200-400 sektorov, 1 má kapacitu 512 B dat
        + zoskupovanie do fyzickýc stránok, alokačných blokov
        + čitanie/zápis – vystavenie hlavy na požadovanú stopu, vyčkanie na natočenie sektoru pod hlavu =>zápis, čítanie
  + Priamy prístup – čítanie/zapisovanie dát na disk v ľubovoľnom poradí
  + omnoho pomalejší prístup do hlavnej pamäte
  + prežije vypnutie napájania i havárie systému
  + k procesoru sa disky pripájajú pomocou I/O zbernice – EIDE, ATA, SATA, USB, ...
  + prenos dát ovládajú radiče
    - Radič disku – je na počíačovej strane I/O zbernice
    - Pre ovládanie softwarom poskytuje rozhranie
    - Akceptuje príkazy od driveru
    - Zahajuje operácie vystavenia, čítania, zápisu
    - Počíta, kontorluje a pripojuje ochranné kódy
    - Vlasntý diskový pohon ovláda Riadiaca jednotka disku – ma obvykle vstavanú cache pamäť => dáta sa obvykle prenášajú medzi cache a povrchom disku, medzi cache a radičom disku a medzi hlavnou pamäťou PC a radičom disku PC
  + OS bývajú disky adresované ako veľké 1-dimenzionálne polia prvkov nazývané logické bloky/alokačné boly/alokačné cluster/...a tie sú zobrazované na sektory disku sekvenčne
  + Pripojovanie disku k serveru – priamo k PC via I/O port, klasicky IDE/ATA architektúra
    - PATA – skôr označované ako ATA, využíva paralelnú komunikáciu, podobné je IDE /EIDE/ATAPI
    - SATA – modernejšia verzia klasického ATA
    - SCSI – drahé, spoľahlivé, výkonné => servery
    - SAS
    - FC – primárne pre sieťové pripojenia
    - FATA – lacnejšie FC
  + pripojovanie disku prostredníctvom siete
    - Priame pripojenie = DAS
    - NAS
      * Pamäťové zariadnie poskytujúce službu File Serveru v LAN sieti ktorá je odolná voči výpadkom
      * Prevádzkový model: klient-server; protokol : NFS/CIFS/SAMBA
      * Malá rychlost a veľká latencia, zraniteľnosť
    - SAN
      * FC technológia
      * Vysokorýchlostná sieť, veľká pamäťova kapacita, odolnosť voči výpadkom, dokonalé zálohovanie a obnova
    - iSCSI – štandard zasielania SCSI dátových blokov prokolu TCP/IP
  + výkonnostné miery disku
    - Access time = doba prístupu = doba ktorá uplynie od vydania žaidosti na čítanie/zápis do zahájenia prenosu dat. Skladá sa z :
      * Doby vystavenia = seek time, snažíme sa ju minimalizovať pomocou tzv plánovanej činnosti disku
        + FIFO-spravodlivé voči procesom
        + SSTF = shortest seek time first – hrozba starnutia
        + SCAN=výťah – ramienko sa pohybuje z okraja do stredu a späť a spravuje požiadavky podľa poradia stop
        + C-SCAN = jednsmerný výťah – ramienko sa pohybuje z okraja do stredu a späť bez spravovania požiadavok
      * Rotačného oneskorenia
    - Rýchlosť prensu dat = rýchlosť, ktorou ide dáta načítať/uložiť
    - MTTF = Mean time to failure = priemerná doba do zlyhania(3-5rokov)
  +  príklad : Magnetický disk s 5 kotúčmi má na kotúči 2 048 stop, stopa má 1 024 sektorov, sektor 512B dat. Kapacita disku je .... 512B x 1024 sektorov/stopu = 0.5MB/stopa .... 5x0.5 MB/stopu x 2048 stop/kotúč = 5GB

* U disku je cieľ čo najrýchlejší prístup a čo najväČŠia šírka pásma
* Disk sa formátuje – 1. fyzické formátovanie = delenie disku na sektor
* 2. Logické formátovanie = vytváranie súboroveho systému

RAID = Redundant Arrays of Independent Disks

* Pôvodne Inexpensive disks, mali nahradzovať veľké drahé disky no teray to sú disky s veľkou kapacitou, spoľahlivosťou a rýchlosťou
* Dáta vieme v RAID rozmiestňovať technikou prúžkovania=striping
* Redundantná disková kapacita sa používa na uchovanie paritnej informácie- zaisťuje obnoviteľnosť alebo zdvojovanie dát
* Najpoužívanješie : RAID1, RAID5 + kombinované RAID1 + 0, RAID 0+1
* Zvýšenie výkonu sa deje prúžkovaním
  + bit-level striping – už sa nepoužíva
  + block-level striping
  + disk striping- dáta sa rozložia na dielčie dat.jednotky a tie sa rozdelia na disky
* Zvýšenie spoľahlivoti sa deje redundanciou – doplnková inf pre obnovu pri poruche
  + mirroring, shadowing
    - Každý disk sa duplikuje (1log.disk je tvorený 2 fyzickými)
    - Každý zápis sa prevedie na obpch diskoch, číta sa z jednoho
    - Ak sa jeden pokazí, číta sa z druhého; k strate dat dôjde pri výpadku oboch diskov
  + duplexing
  + block interleaved parity
* RAID 0
  + žiadna redundancia, žiadna odolnosť voči poruchám, výpadok 1disku znamená stratu všetkých dat
  + v aplikáciách ktoré majú vysoké požiadavky na výkon a strata dát nie je kritická
* RAID 1
  + redundancia sa dosahuje duplikovním všetkých dat
  + vhodné pre kritické aplikácie z hľadiska straty dát
* RAID 2
  + Error-Correcting Code, netypické
* RAID 3
  + Bit-Interleaved Parity, netypické, plne nahradené RAID5
* RAID 4
  + analógia RAID 0 , doplnený je disk s paritnou informáciou
  + čítanie bloku – číta sa 1 blok s dátami
  + čítanie bloku vypadnutého disku – číta sa daný blok zo všektých diskov vč. Paritného a hodnota sa doráta pomocou XOR
* RAID 5
  + veľmi populárna implementácia RAID
  + minimum sú 3disky – prekladané data + parita
  + zvládne výpadok 1 disku
* RAID 6
  + datové disky s dvojitou paritou, minimáne 4disky
* RAID 0+1
  + striping na báze RAID 0 a nad ním zrkadlenie na báze RAID 1
  + RAID 1 = bezpečnosť a RAID 0 = výkon
  + výpadok viac než 1 disku je tolerovateľný ak k nemu dôjde v jednej sade RAID 0
  + pre aplikácie požadujúce výkon i spoľahlivosť
* RAID 1+0
  + Zrkadlenie na báze RAID 1 a nad ním stiping na báze RAID 0
  + lepší výkon než RAID5
  + môže vypadnúť disk z každej sady RAID1
  + aplikácie požadujúce výkon i spoľahlivosť



SSD = Solid-state drive

* Neobsahuje pohyblivé mechanické časti
* Sú nehlučné, nižšia spotreba, ľahšie, odolnejšie voči nárazom, rýchlejšie
* Obmedzená životnosť
* Skôr alebo neskôr, SSD zvíťazia nad HDD

TERCIÁRNE PAMATE

* Magnetooptický disk
* Optický disk – nepoužíva magnetizmus, používané v CD-RW a DVD-RW
* Data na RW disku sa dajú modifikovať opakovane
* Data na WORM sa dajú zapśať len raz
  + Zápis sa deje laserovým svetlom kde sa v aluminiu vypáli dierka
  + spoľahlivé,trvácne
* CD – neprepisovateľné, 12cm formát má 60min zvukového záznamu
* CD-ROM – neprepisovateľné, spoľahlivejšie než CD
* CD-R – WORM, archivácia
* CD-RW – podobné ako magnetický disk
* DVD – až 17GB dat pri obojstrannom zázname
* DVD-R – WORM, jednostranný záznam
* DVD-RW- jednostranný záznam
* Blu-Ray DVD – informácia vo vysokom rozlíšení, používa blue-violet laser
* Pásky – vhodné pre archiváciu, uchovávaniu veľkých objemov dat, pomalé, pásky sú append-only
* Rýchlosť u terciárnych pamätí je ovplyvňuje šírka pásma a oneskorenie

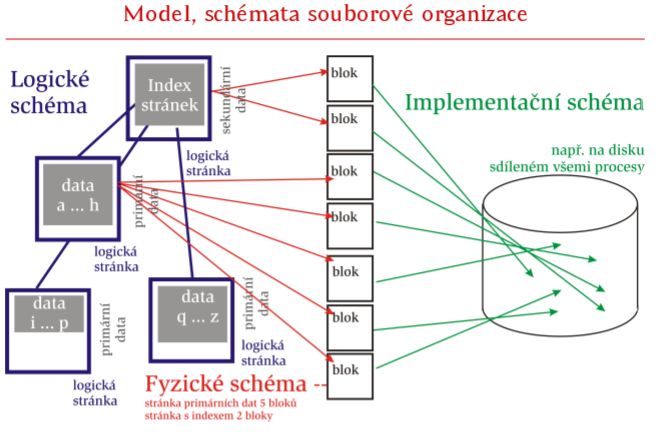
# Súbor, súborové organizácie

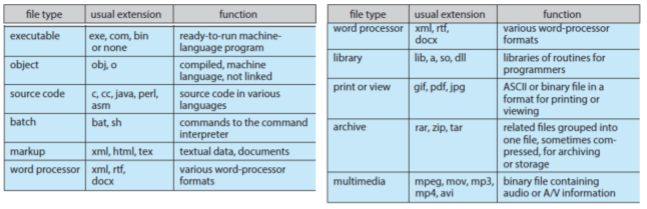
Základný nástroj pre dlhodobé uchovávanie a zprístupňovanie dát na vonkajších pamätiach = databázy

Databáze = kolekcia súvisiacich dat

* Je navrhovaná pre použitie aplikačnymi sytémami
* Býva spravovaná systémom riadenia báze dát = systémom súborov => ten je súčasťou middleware (funkčnej vrstvy medzi OS a aplikačnými systémami)
* V PC (na vonkajších pamätiach) sa uchovávajú ako kolekcie pomenovaných množín dat = súbory, file
  + - Súbor je kolekcia podobných záznamov = records => kolekcia atribútov kt. Charakterizuje istý objekt
      * Logický záznam : kolekcia hodnôt atribútov
      * Fyzický záznam : logický záznam + definícia dĺžiek atribútov
      * Pamäťový priestor pre zobrazenie záznamu vo vonkajšej pamäti = BLOK
      * Pamäťový priestor pre zobrazenie atribútu vo vonkajšej pamäti = POLE
      * Homegénne (hodnoty atribútov sú primitívne typy,je deklarovateľný formou S(A1 : D1, A2 : D2, ...), tj obsahuje zázanmy jedného typu / nehomogénné (hodnoty atribútov nie su primiívne typy)
    - Koncept súboru je voľný - z  pohľadu užívateľa je to najmenjšia adresovateľná jednotka vonkajšej pamäte, môžu obsahovať programy alebo dáta, môžu byť formátované
    - OS poskytuje služby ktoré operujú so súbormi –
      * Konštruktory / deštruktory = create /build/ remove
      * Modifikátroy – vloženie / rušenie / modifikácia položiek záznamu = insert / delete / update (write) (modify)
      * Inšpektory – získanie záznamu zo súboru do vnútorenj pamäte / dotaz / generovanie správy = read / (fetvh)(get) /query (find) / list
      * Údržba súbboru – reorganizácia = open / close
    - 4 pojmy charakterizujúce štruktúru súboru
      * + báza dát = DATABASE = kolekcia vzájomne súvisiacich dát pozostávajúcich z 1 alebo viac súborov napr. Inštitúcia
        + súbor = FILE = pomenovaná kolekcia príbuzných záznamov napr zamestnanci
        + záznam = RECORD = štrukturovaná jednotka, kolekcia polí napr. Údaje o sebe
        + pole = FIELD = elementárna dátová jednotka záznamu napr meno, vek, pohlavie
    - Súbory sa uchovávajú na energeticky nezávislých vonkajších pamätiach
    - Správu zabezpečuje systém súborov = file system
    - Pre manipuláciu s dátami musia byť subory prístupné ako celky => to riešia adresárové služby no rovnako efektívne musia byť prístupné aj záznamy súborov => to rieši súborová organizácia (YAAAY, konečne názov predmetu)
* Štrukturálne najzákladnejším prvkom dát je pole (field) – z hľadiska priestoru, syntaxe ALE z hľadiska obsahu to je položka resp atribut, vlastnosť
  + - S každým atribútom súvisí datový typ určujúci obor možných hodnôt atribútu a množinu operácií nad ním (integerm long, Boolean, string, Date...)
    - Atribút môže mať fixnú/ premenlivú dĺžku
    - Pomenovaný /nepomenovaný
    - Kolekcia atribútov a ich definície vytvárajú typ záznamu

SYSTÉM SÚBOROV = FILE SYSTEM

* Musí
  + umožniť užívateľom/procesom manipuláciu s dátami
  + každý u/p môže vytvárať/mazať/meniť/ ... súbory
  + každý u/p môže riadeným spôsobom zprístupňovať súbory iných u/p
  + každý u/p môže riadiť povolené prístupy k jeho súborom
  + Každý u/p môže reštrukturalizovať svoje súbory
  + každý u/p môže svoje dáta zálohovať a obnoviť
  + každý u/p môže identifikovať svoje súbory symbolickými menami
  + zaistiť , že dáta sú validné
  + eliminovať stratu dát
  + optimalizovať výkon z hľadiska systému i používateľa
  + podporovať na rôznych zariadeniach definované API
  + umožniť súbežnú činnosť viacrých u/ aplikácií
* Cieľom návrhu súborového systému je minimalizáca počtu prístupu na disk, ideálne 1 prístup = 1 operácia
* Trocha histórie
  + - Do 50.-60. Rokov XX storočia boli súbory len na páskach = sekvenčný prístup
    - 60. – 70. Roky – objav disku a s ním aj možnosť priameho prístupu k zázanmom
      * Sprístupňovaný záznam je determinovany kľúčom a priamy prístup sa deje hašovaním - pomocou indexov tvorených dvojicami { hodnota kľúča, adresa záznamu pamäti v súbore}
      * Potom objavený strom a teda možnosť optimalizácie hľadania záznamu z O(n) na ( log n) – good idea ale malo to aj svoje mínusy - > riešenie ? insert a delete automaticky vyvážajú strom => B-stormy / B+ stromy / statické hašovanie / dynamické haš.
* Metódy oddeľovania záznamov v súboroch
  + fixná dĺžka záznamu
  + premenlivá dĺžka záznamu: oddelenie explicitne (napr hodnota vždy rovnaká na začiatku zoznamu) / oddeľovač / ukazateľ
  + fixná štruktúra položky
  + variabilná dĺžka položky : viď záznam
* Blokovanie záznamov
  + blok = jednotka dat z pohľadu I/O , má dĺžku B
  + neblokovaný záznam – cieľom je 1 IO operácia / 1 záznam, blok obsahuje práve 1 záznam
  + blokovaný záznam – blok obsahuje / „blokuje“ celistvý počet záznamov
  + blokovací faktor b = [B/R] (zaokrúhlené dole) kde R je dĺžka záznamu
  + separované záznamy – časté riešenie pre súbory so záznamami premenlivej dĺžky
  + kľúč = atribúty identifikujúce záznam, poznáme aj relativný kľúč ktorý je zvláštným typom kľúča udávajúcim pozíciu záznamu vzhľadom k počiatku súboru
    - Primárny kľúč identifikuje záznam jedinečne, nemal by obsahovať aplikačne závislé data = dataless key
    - Sekundárny kľúč = jeden z mnohých, nutne neidentifikuje jednoznačne
* Dotaz nad súbrom
  + totálna vyčísliteľná funkcia definovaná na univerze súboru napr. dotaz= FETCH (súbor, kľúč) / odpoveď =záznam alebo kolekcia záznamov
  + jendorozmerný dotaz / viacrozmerný dotaz (na úplnú zhodu / na čiastočnú zhodu...)
* Hierarchia suborovej organizácie
  + LOGICKÁ SCHÉMA
    - Logická pamäť so štruktúrou optimalizovanou tak, aby efektívne vykonávala operácie nad záznamami
    - Člení sa na logické stránky
    - Obsahuje primárny súbor = aplikačné data ale aj sekundárne súbory = indexy, registre
    - Záznamy oboch týchto súborov môžu byť v logických stránkach blokované – blokovací faktor λ = skutočný počet záznamov v LS / možný počet
  + FYZICKÁ SCHÉMA
    - Zobrazenie logických stránok do fyzickych stranok podľa typu vonkajšej pamäti
  + IMPLEMENTAČNÁ SCHÉMA
    - Alokácia fyzických stránok v použitom zariadení
    - Zložitosť : preistorová – potrebný objem fyzických stránok pre zobrazenie súboru a časová – počet V/V operácií s fyz.stránkami, počet načítaných fyz.stránok do RAM, počet zapisovaných fyz.stránok do zariadenia
* Typy súborov
  + podľa obsiahnutých dát v súbore sú napr. Numerické, textové, binárne
  + typyicky sa deklarujú príponou .txt, .xls, .doc, .exe, .gif, .mpeg, ...

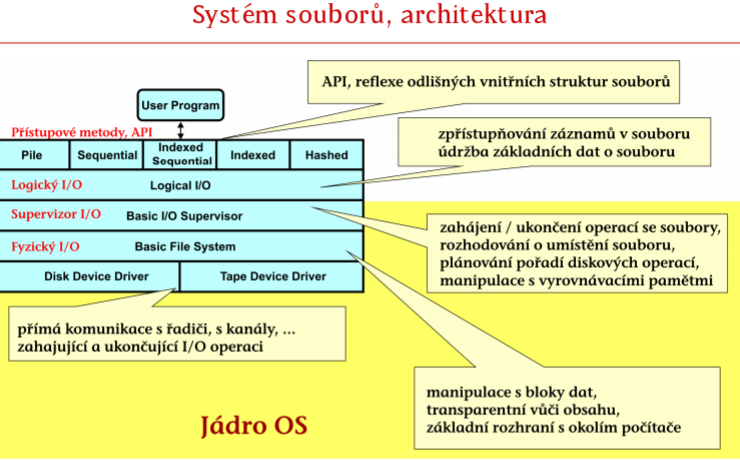


* Organizácia súborov môže byť relatívne voľná ale aj formátovaná
  + Z pohľadu OS môže ísť o postupnosť bajtov (uplatňuje UNIX, Windws) alebo o postupnosť záznamov alebo o strom záznamov
* Niektoré kritériá na ideálny súborový sytém môžu byť protichodné :
  + minimalizácia pamäťovej zložitosti => minimalizácia redundancie
  + minimalizácia doby prístupu => napr.duplikáciou dát ale vedie k zvýšeniu redundancie
* Ako sme už vraveli , cieľom je minimalizácia prístupu no sú tu aj iné ciele – vkladanie, modifikácia a rušenie transakcií na záznamoch majú byť rychlé , jednduché a nesmú porušovať ostatné záznamy, takisto tieto operácie nesmú spôsobiť duplikáciu záznamov a tieto áznamy majú byť uložené tak, abz bola cena potrebnej pamäte minimálna
* Poznáme viacej metód organizácie
  + sekvenčný prístup
    - nedá sa skákať, vieme iba prejsť na začiatok súboru alebo ďalší záznam
    - Aplikovateľná na páske/disku
    - Zložitosť je O(n)
  + priamy prístup –
    - zprístupnenie v ľubovoľnom poradí, DBS!!
    - Určenie miesta záznamu vieme dosiahnuť pomocou indexu alebo hašovania
      * Určenie miesta pomocou indexu
        + Index ako sekvenčná štruktúra usporiadaných ukazateľov na záznamy (pole, tabuľka, zoznam dvojic)
        + Index ako stromová štruktúra usporiadaných ukazateľov na záznamy (BST, B+ tree, B tree)
      * Určenie miesta pomocou hašovania
        + algoritmicky vieme odvodiť ukazateľ záznamu z hodnoty kľuča
* Poznáme viac foriem organizácie súborov, základných je 5
  + hromada = HEAP
    - Data sa ukladajú podľa poradia vloženia
    - Nehomogénny súbor, môže obsahovať záznamy premnnej štruktúry
    - Cieľom je uložiť veľa dát
    - Malá využiteľnosť, search = O(n)

* + sekvenčný súbor = SEQUENTIAL FILE
    - záznamy majú pevný formát, mená a obory hodnôt sa označujú ako vlastnosti
    - polia ktoré obsahujú kľúče = kľúčové polia
  + Indexovaný sekv.súbor = INDEX-SEQUENTIAL FILE – vyhľadávanie indexom
    - Záznamy sekv. Súboru sú usporiadnaé podľa kľúča a sú doplnené o index a prietokovú oblasť
  + indexovaný súbor = INDEXED FILE – súbor doplnený viac indexmi
  + skladba atribútov záznamu je známa
    - Súbor nie je usporiadaný
    - Index môže mať tabuľkovú ale aj hierarchickú štruktúru B-strom
  + hašovaný súbor = HASHED FILE / súbor s priamym prístupom = RANDOM ACCESS FILE
    - Skladba atribútov záznamu je známa
    - Lokalizácia hľadaneho záznamu sa odvodí algoritmicky, hašovacou funkciou s kľúčom
    - Optimálna voľba pre aplikácie s kritickým faktorom doba prístupu

# Súborové systémy - koncepty a rozhrania

* Základný prístup k vonkajšej pamäti poskytuje OS cez systém súborov
  + umožňuje vytvárat kolekcie dát
  + súbory dlhodobo uchovávať i po uzavrení užíateľskej relácie
  + pomenovávať súbory
  + štruktúrovať súbory tak, aby vyhovovali aplikácii
  + rieši správu zdieľaných zdrojov
* Pre väčšinu užívateľov je to najviditeľnejší prejav OS
* Tvoria ho 2 komponenty
  + kolekcia súborov
  + adresárová štruktúra
* Nemusí byť plne implementovaný v OS, niektoré funkcie môžu vystupovať ako systémové programy



* Atribúty súboru udržované súb. Systémom
  + meno – jediná cielene udržovaná informácia vo forme čitateľnej človekom
  + typ
  + Organizácia
  + umiestnenie
  + rozmer
  + ochrana
  + čas, datum, história ....
  + tieto všetky sú uchovávané v adresárovej časti systému súborov – musí bť uložená na energeticky nezávislej časti pamäte = disk
* FYZICKÝ x LOGICKÝ súbor
  + fyz súbor :
    - Vonkajšia pamäť, je pomenovaný vonkajším (text) menom súboru, manipuluje s ním OS, jeho existencia je zaznamenaná v adresári súborov
  + log súbor :
    - Rúra z/do ktorej proces získava/dodáva data, je pomenovaný vnútorným (numerické, prideľuje sa otvor.súborom) menom súboru, logicy súbor je zviazaný s fyzickým súborom OS službou =>otvorenie súboru
* Pri otváraní súbru OS vyhľadá záznam o súbore Fi v adresári a presunie ho do operačnej pamäte PC do tabuľky otvorených súborov (close funguje opačne)
* Štandardné log. Súbory : stdin /stdout / stderr, dajú sa presmerovať na fyzický súbor (príklady kusov kódu v C/C++ na prácu s FILE ako fopen, fclose 17-23 <https://www.fi.muni.cz/usr/staudek/vyuka/filesys/05_rozhrani_sos.pdf> )
* Správa otvorených procesov vyžaduje istú údržbu dát – v tabuľke ovorených súborov procesu sa udržiava ukazateľ na práve sprístupňovaný záznam, prístupové práva a ukazateľ na tabuľku otvorených súborov systému

ADRESÁRE

* Kolekcia dát obsahujúcich informácie o súboroch uchvávaných na disku – adresár je sám súborom
* Logicky sa organiyujú tak, aby sa dosiahla efektívnosť, zrozumiteľné pomenovanie pre užívateľa (2 users môžu rovnako pomenovať rôzne súbory alebo rôzne 1 zdieľaný súbor), ľahké zoskupovanie, ...
* Pomocou nich vieme súbory organizovať a jasne separovať na báze vlastníctva, aplikácií, prípadne ďalších kritérií => vylepšenie bezpečnosti, integrity, ...
  + 1-úrovňový adresár
    - Jediný adresár v celom systéme
    - Nerieši sa problém unikátnosti pomenovania súborov ani problém zoskupovania
    - Súčastnosť : v telefónoch, kamer=ach, prehrávačoch
  + 2-úrovňový adresár
    - Separácia užívateľov
    - Súbory vieme pomenovať prístupovou cestou
    - 2 useres môžu rovnako pomenovať rôzne súbory
    - Efekívne hľadanie, nerieši problém zoskupovania
  + hierarchický adresár, adresár so sromovou štruktúrou
    - Efektívne hľadanie hodnoty v uzli
    - Rieši nezávislé pomenovanie i zoskupovanie
    - Pojem pracovný adresár = >cd/spell/mail/prog
    - Pojem absolútna / relatívna prístupová cesta = prístupová cesta začínajúca v koreni / prac.adresári
    - Súbor vieme vytvriť / zmazať v pracovonom adresári new/rm alebo kdekoľvek doplnením prístupovej cesty pred file name
  + acyklický adresár
    - Umožňuje zdieľat adresáre a súbory, nie je strom
    - Aliasing = jeden objekt má 2 rôzne mená
    - Pri prechádzaní adresáru nedojde k zacykleniu lebo sa povolí vytváranie väzieb iba na súbory, nie adresáre
* Pripojovanie adresárov sa deje cez File System Mounting

Dobré praktiky organizovania súborov :

* Má byť dôsledne dokumentovaná
* Pravidlo 5 C = clear, concise, consistent, correct, conformant
* Zmysluplná hierarchia a pomenovania, nepoužívať generické mená

Zdieľa nie a ochrana súborov

* Typická implementácia zdieľania je vedená cez nástroje ochrany
  + Voliteľné riadenie prístupu = DAC
    - Vlastník (ten kto súbor vytvoril) má možnosť určovať co sa dá so súborom robiť a kto to môže robiť
    - Typy prístupu : read / write / execute / delete / list
    - Najviac rozšírený nástroj – Network File System = NFS
  + Povinne riadenie prístupu = MAC
    - Určovanie čo sa dá robiť so súborom a kto to môže
    - Politiku riadenia prístupu definujú pravidlá implementované v systéme ako súčasť bezpečnostnej politiky
    - Subjekty v sytéme nemajú možnosť pravidlá meniť
* UNIX – model voliteľnej ochrany – read/write/execute, user/group/others ... príkaz chmod napr. chmod 751 game
* Riadenie prístupu implementované cez : maticu povolených prístupov, Access control list = ACL , Capability tickets

Múdrosť dňa (slide 50, lecture 5)

„Nie je nič podozrelejšie než keď ti niekto chce urobiť dobre a zdanlivo za to nič nechce“

* V kontexte toho že OS sa niekedy snaží uživateľovi uľahčiť život a teda vkladá pomocné oddeľovače do súborov ale to nie je dobré
* Príklad systému súborov – UNIX, typy súborov
  + riadny súbor – dáta vytvarané užívateľmi, aplikáciami, programami...
  + adresár – zoznam mien súborov + ukazatele na reprezentáciu súborov
  + pomenované rúry – komunikačný nástroj medzi procesmi formou toku slabik
  + väzba
  + symbolická väzba
  + charakteristiky systému súborov v UNIXe :
    - Stromová štrutúra
    - Každý súbor vieme jednoznačne identifikovať absolútnou cestou
    - Koreň stromu reprezentje znak „/“
    - . => tento adresár ; .. => nadradený adresár
  + stdout / stdin /stderr
  + presmerovanie < / >
  + rúra – prog1 | prog2 (stdout z prog1 je stdin pre prog2
  + príkazy : cat, mv, rm, chmod, ls, mkdir, ...
  + Inode = riadiaca štrutúra obsahujúca kľúčové inf. Potrebné pre OS pri správa istého súboru. Každý súbor je riadneý práve jedným inode
  + adresár v UNIXe je súbor obsahujúci zoznam mien súborov + ukazatele na reprezentáciu súborov

Windows, NTFS

* Systém súborov pre pracovné stanice a servery
* Kľúčové rysy : obnoviteľnosť po výpadku, bezpečnosť, aplikovateľnosť na veľmi veľké disky a súbory (>>FAT), ....

Remote File Systems

* Pre sprístupneňovanie medzi systémami používa sieťové služby
  + manuálne – cez ftp programy
  + automaaticky – distribuovaným súborovým systémom
  + poloautomatika – cez World Wide Web
  + klient pri aplikácii modelu klient-server si môže pripojovať vzdialené súbory uchovávané na serveroch – štadardný protokol je NFS pre UNIX, pre Windows CIFS
  + unifikovaný prístup k informáciám potrebný pre vzdialené počítanie implementujú distribuované IS pomocou distribuovanej pomenovacej služby typu :
    - LDAP
    - DNS
    - NIS

# Súborové systémy – implementácia

Štruktúra systému súborov – logická pamäťová jednotka a kolekcia súvisiacich informácií

* Systém súborov pôsobí na vonkajšej pamäti, hierarchické usporiadanie
* Logical File System – správa metadat
* File Organization module – správa (voľnej) pamäti na disku
* Basic File System : čítanie /zápis fyz.stránok z/na disk
* I/O Control : drivery, správa prerušenia
* Vytváranie súboru
  + aplikačný proces volá logický FS a zadá parametry vytvorenia súboru. Ten vytvorí FCB na disku opraví adresár a FCB uloží na disk
  + s existujúcim súborom môže proces pracovať iba po jeho otvorení
* Datové štruktúry implementácie súb.sys.
  + otváranie súboru – otvorením sa filename zmení na vonkajšie meno súboru, mapuje na index, vnútorné meno súboru. Unix – file descripor, Windows – file handle
  + čítanie súboru – súbor sa identifikuje indexom, vnútorným menom súboru
* Čistý disk x Disk spravovaný súb.sys.
  + Raw disk – aplikácia sama manipuluje s pamäťou na disku
  + Disk spravovaný OS – členenie priestoru na oblasti = partitions napr. root partition ktorý obsahuje závadzanú kopiu OS.
* OS sa virtualizujú – cieľom je možnosť používať jendotné API i pre odlišné typy súborových sytemov

Implementácia adresárov – typické riešenie

* Lineárny zozna mien súborov s ukazateľmi na bloky dat
  + jednoduché na naprogramovanie
  + spracovanie dotazu je lineárne zložité pri neusporiadani adresára napr FAT a pri usporiadaní adresára jee logaritmicky zložité
* Komplexná dat šturktúra napr B+ strom – SEARCH - log.zložtosť, NTFS
* Hašovaná tabuľka – SEARCH- konštantná zložitosť, môžu sa vyskytovať kolízie

Metódy prideľovania pamäte súborom

* Prideľovanie súvislých diskových priestorov
  + každý súbor zaberá množinu susedných blokov
  + jednoduchá metoda, možný sekvenčný aj priamy prístup
  + Priestorová neúspornosť – hrozba externej fragmentácie
  + súbory nemôžu rásť
  + varianta Extent-Based File System
* Viazané prideľovanie disk.priestoru – tzv mapa disku = File Allocation Table(FAT)
  + súbor je viazaným zoznamom nesusedných blokov disku
  + jednoduchá metoda vhodná len pre sekvenčný prístup
  + ľahká expanzia súboru – metoda používaná pre správu voľnej pamäte
* Indexované prideľovanie disk.priestoru – mapa súboru
  + Je potrebná tabuľka, indexový blok
* Alokačný blok = cluster – postupnosť susediacich sektorov
  + častokrát je problém s jeho dĺžkou – krátky cluster = fragmetácia súboru lebo sa alokujú nesusedné clustre / dlhý alokačný blok = hrozí vnútorná fragmentácia

súborový systém FAT

* Základné verzie FAT sú vhodné pre disky s kapacitou do 0,5GB
* Vlastné dáta sa čítajú z cleej plochy disku, ale informácie o pozíciach clusteru z FAT na počiatku disku
* FAT32
  + zvýšenie možného počtu alokačných záznamov na 232 z 216 no v reále 228
  + efetívnejšie využitie miesta na disku

súborový systém NTFS

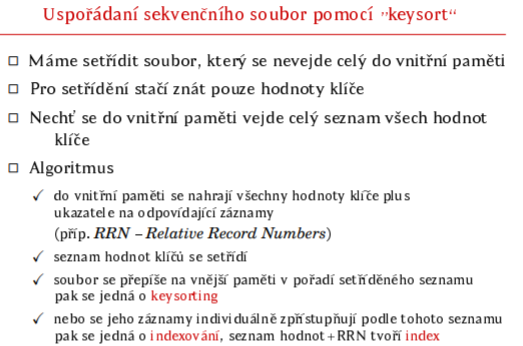
* Základnou štruktúrou je NTFS zväzok ktorý je obsiahnutý v diskovej oblasti = partition
* Štruktúra zväzku:
  + ID sektor – Boot
  + tabuľka MTF obsahujúca zákldné info o všetkých súboroch a adresároch na zväzku
  + oblasť pre uloženie dat súborov, ktorá sa nevošla do MFT
  + každý zväzok obsahuje súbory, adresáre, bitmapy, ... a každý je organizovaný ako postupnosť alokačných blokov (cluster)
* Vnútorné usporiadanie
  + diskové adresy NTFS sú poradové 64bit čísla = logical cluster numbers
  + súbor v NTFS je
    - štrukturovaný objekt tvorený atribútmi (pomenované /nepomenované)
    - definovaný jedným alebo niekoľkými záznamami v poliach kt sú uchovávané v tabuľke => MFT
    - zobrazený pomocou atribútov (rezidentné-v MFT/nerezidentné-v externých alok.blokoch)
    - každý súbor má
      * vonkajšie meno (sú v hierarchii – B+strom)
      * jedinečné vnútorné meno, ID, file reference
      * security descriptor ktorý obsahuje príznak prístupových práv = access token a zoznam prístupových práv = ACL
* MFT
  + je hlavná dát.štruktúra na NTFS zväzku – definuje obsah zväzku
  + je lin postupnosť 1KB záznamov pevnej dĺžky
  + pre každý súbor existuje aspoň 1 záznam
  + teda ide vlastne o relačnú DB; záznam MTF obsahuje meno súboru, časové razítko, bezpečnostný deskriptor, data, ...
  + voľné a obsadené MFT mapy sú evidované bitovou mapou
  + pridávabím zväzkov sa mapa rozširuje no rušením sa len ponechávajú voľné
* NTFS Recovery
  + všetky korekcie sú transakcie, o ktorých je vytváraný protokol (log) – predtým než sa niečo zmení transakcia zapíše záznam do protokolu ktorý obsahuje REDO aUNDO, po zmene sa poznačí COMMIT RECORD. AK systém padne, vráti sa všetko späť podľa protkolov
  + nie je záruka obnovy všetkých užívateľských súborov
* NTFS podporuje disk mirroring ako schému ktorá udržuje v dvoch rovnakých oddieloch na dvoch diskoch identický obsah = mirror set
* Tiež podporuje koncept RAID 5

NTFS X FAT

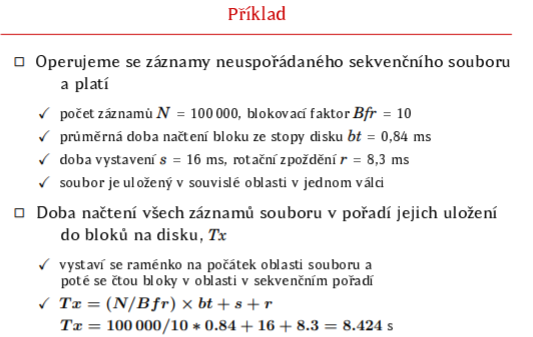
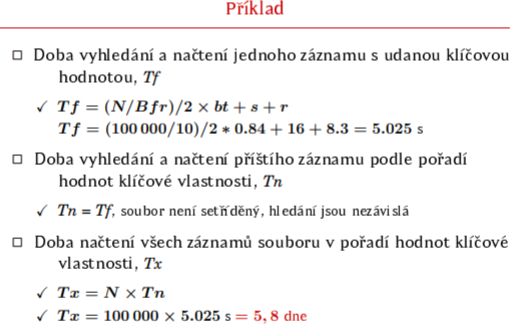
* NTFS je určený pre disky s kapacitou >500MB
* FAT nemá bezpečnostný deskriptor
* Podporu obnovy má len NTFS
* B+ stromová štruktúra NTFS – rýchlejší prístup k súborom, minimalizácia prístupu na disk

# Sekvenčné súbory

* Neusporiadaný
  + homogénny
  + Iba sekvenčný prístup
  + INSERT O(1), FIND O(N), pri blokovacom faktore b O(N/b) – blokovanie neovplyvňuje zložitosť, ovplyvňuje efektivitu využivania pamäte
* Usporiadaný
  + podľa nejakého kľúča
  + INSERT
    - Bezprostredné ukladanie dát do primárneho súboru + in-line reorganizácia
    - Občasná dávková reorganizácia
    - Udržovanie poradia formou reťazovej štruktúry
    - Keysort – uržuje sa netriedený primárny súbor + triedený index
  + SEARCH O(log2N) teda binarne vyhľadávanie



KEYSORT <https://www.fi.muni.cz/usr/staudek/vyuka/filesys/07_sekvencni_soubory.pdf> 6-10

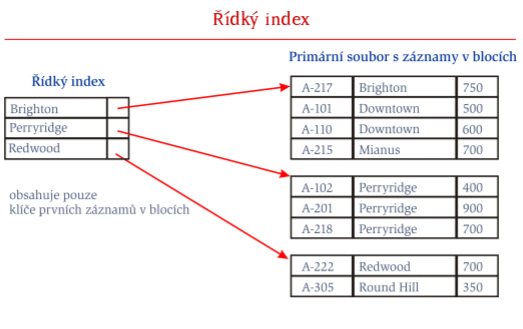
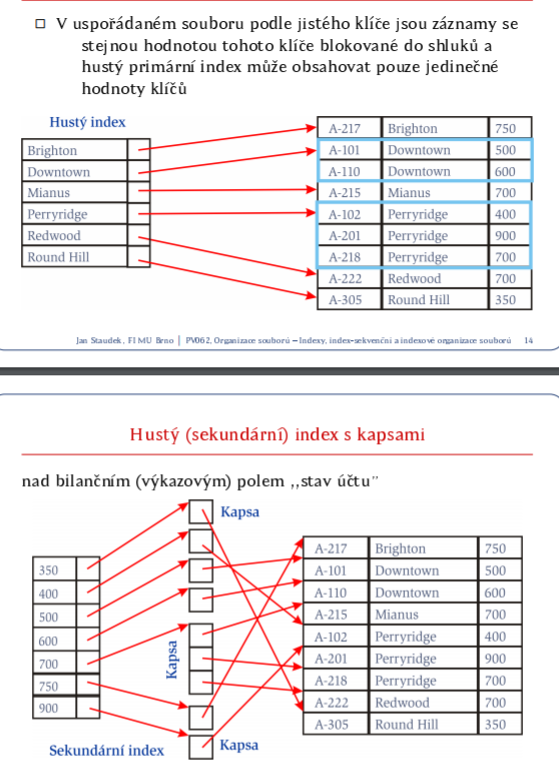


# Indexy, index-sekvenčné a indexové organizácie súborov

Rýchlosť reakcie na položený dotay patrí medzi hlavné problémy pri návrhu databáz. Sekvenčné prechádzanie jednotlivých záznamov v DB je neefektívne. Pre urýchlenie riešenia dotazov sa k DB doplňuje pomocná štruktúra – index.

Index – obvyklá forma {vyhľadávací kľúč, ukazateľ záznamu}

* Implementácia súboru s indexom
  + primárny súbor – indexované data
  + sekundárny súbor – index , môže ich byť viacej
* Základné techniky implementácie indexov
  + lineárne indexy – tabuľka dvojic usporiadaná podľa hodnoty kľúča
  + indexové bitové mapy
  + hašované indexy
  + stromové indexy
* Základné typy indexov
  + jednoúrovňové
  + Hustý
    - Indexový záznam pre každú hodnotu kľúča napr. ID
    - Indexový záznam môže odkazovať na kapsu ktorá môže obsahovať
      * Ukazatele na všetky záznamy s danou hodnotu konkrétneho vyhľadávacieho kľúča alebo
      * Všetky záznamy v primárnom subore zotriedené podľa vyhľadávacieho kľúča
    - býva usporiadaný
  + riedky
    - Indexový záznam pre vybrané hodnoty kľúča tzv. Separátory intervalov hodnôt vyhľadávacieho kľúča, vojde sa do RAM
    - Umožňuje efektívnejšie riešiť INSERT, DELETE
    - Idea SEARCH – v indexe nájdeme najväčší menší kľúč než hľadaný a potom prehľadáme daný interval
  + Primárny
    - Pomocou primárneho indexu je uporiadaný primárny súbor
    - Je vždy jediný, neobsahuje duplicitné hodnoty
    - Býva riedky, no môže byť i hustý
  + Sekundárny
    - Môže ich byť viac, môže obsahovať duplicitné hodnoty
    - každý iný index k primárnemu súboru než primárny index
    - musí byť hustý
  + viacúrovňové indexy
    - Ku každému indexu prvej úrovne vieme vypracovať riedky index druhej úrovne

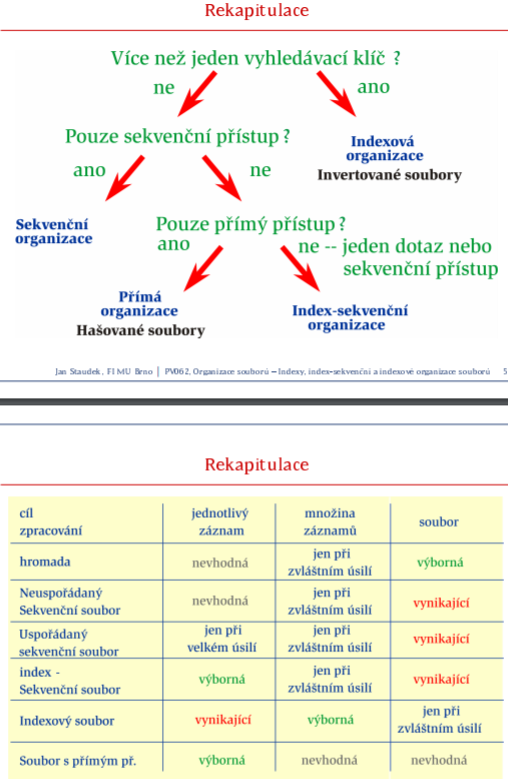


Indexované súbory

* Primárny súbor s dátami je obvykle netriedený
* Vypracujú sa indexy pre všetky možné/používané dotazy
* Riadne indexy musia byť husté
* Priame a nepriame indexovanie pomocou sekundárnych indexov
  + priame – sek.index viaže s hodnotou kľúča ukazateľ záznamu v primárnom súbore
  + nepriame – sekundárny index viaže s hodnotou kľúča hodnotu primárneho kľúča

Index-sekvenčné organizácie súborov

* Bázové komponenty v schéme súboru :
  + primárny súbor dát – zotriedený podľa kľúča
  + indexová štruktúra
  + oblasť pretečenia - pre dočasné riešenie anomálií (na každom valciˇ, pre INSERT)
* Štruktúra indexu
  + riedky index, indexujú sa bloky obsahujúce záznamy
  + hierarchické usporiadanie
* Použitie iba operácie FETCH – statický súbor
* Operácia INSERT – dynamický súbor
* Oblasť pretečenia pre riešenie operácie INSERT no lepšie je periodicky prevádzať operáciu Reorganizácie
* Indexované oblasti v primárnych dátach – stopy, valce, zväzky
* Problémy :
  + vloženie záznau môže byť obtiažna operácia, zrušenie záznamu môže zanechať dieru, so zvyšujúcim sa počtom záznamov výkon zprístupňovania klesá
* Rekapitulácia :
  + index je samostatný, usporiadaný sekundárny súbor v schéme organizácie súboru
  + pre primárny súbor s >= 106 záznamami index nevieme racionálne umiestniť do RAM
  + 🡺 index sa teda nahradí hašovacou funkciou alebo do B stromu



# Hierarchické indexy, B stromy, B+ stromy

Cieľom je vyriešiť dotazy niekoľko málo operáciami bez hašovania

Indexovanie – mechanizmus pre riešenie odpovede na dotaz zisťujúci hodnotu záznamu ktorého kľúč vyhovuje podmienke

Indexy na báze vyhľadavacích stromov – sú alternatívnou organizáciou voči index-sekvenčnej organizácii súborov. Obe organizácie zefektívňuju riešenie dotazov nad súborom ALE....

GRAFY!!!

Graf G = (V, E)

* V – konečná neprázdna množina uzlov
* E – množina hran prepojujúcich uzly y V
* Hrana, orientovaná hrana, orientovaný graf – dvojice , kde v a w sú prvky V
  + orientovaná hrana – usporiadaná dvojica uzlov (v,w)
  + orientovaný graf – graf s orientovanými hranami
* Neorietovaný graf – ku každej hrane (v,w) existuje i hrana (w,v) tzv. Obojstranne orientovaná hrana
* Susedné uzly – v je sused s w ak v grafe existuje hrana (v,w)
* Stupeň uzlu – počet hran ktoré s uzlom incidujú
  + u orientovaných grafov rozlišujeme vstupný a výstupný stupeň
* Arita – počet výstupných hran z uzlu
* Cesta – postupnosť uzlov, mezi ktorými vedie postupnosť hran
* Cyklus
* Smyčka
* Jednoduchá cesta
* Ohodnotený graf – váha, ohodnotené hrany – kvantitatívne ohodnotenie vzťahu
* Súvislý graf
* Acyklický graf – žiadna cesta nen´cyklom
* Jednoduchý graf
* Strom – neorientovaný súvislý acyklický graf čiže graf bez cyklu
* Koreňový strom
  + orientovaný strom ktorého uzly majú vstupný stupeň rovný jednej až na koreň
  + uzly sú radené do vzťahu typu rodič-potomok
  + koreň nemá rodiča
  + uzol bez potomka je llist
  + uzol ktorý má aspoň 1potomka a neni koreň je vnútorný uzol
* Strom - charakteristiky
  + hĺbka uzlu = dĺžka cesty vedúcej od koreňa k listu(koreň=0)
  + úroveň stromu – množina vrcholov s rovnakou hĺbkou(koreň=0)
  + hĺbka stromu – počet úrovní stromu
  + šírka stromu – počet uzlov na rovnakej úrovni
  + výška stromu h=K-1 => maximálna hĺbka stromu; iba koreň má výšku=0
  + vyvážený strom – vyvežovanie je rovnomerné rozkladanie uzlov v úrovniach
  + usporiadaný strom – priamy potomkovia každého uzlu sú zoradení
* Binárny strom
  + konečná množina uzlov ktoráq je buď prázdna alebo obsahuje koreň a ľavý a pravý podstrom
  + Rôzne stromy a aplikácie <https://www.fi.muni.cz/usr/staudek/vyuka/filesys/09_hierarchicke_indexy.pdf>

21-27

* BVS
  + uzly sú usporiadané tak, aby bolo možné rýchlo vyhľadávať danú hodnotu reprezentovanú uzlom – kľúč
  + každý uzol má kľúč, ľavý podstrom obsahuje iba kľuče menšie než hodnota kľúča v parent node, pravý podstrom iba kľúče väčšie než kľúč v parent node
* B strom
  + Není binaŕny ani balanced !!! ale B = Bayer & McCreight tree => Bayerov strom
  + Listy sú na rovnakej úrovni
  + Koreňový strom s istým počtom vnútorných uzlov a listov
  + každý uzol má viac než jeden kľúč, sú usporiadané
  + existuje podstrom uzlu obsahujúci kľuče menšie/väčšie než najmenší/najväčší kľúč v uzle
  + pre každý susedný pár kľúčov existuje podstrom ktorý obsahuje kľúče z intervalu medzi týmito kľučami
  + uzol obsahuje počet ukazateľov o jeden väčší než je počet kľúčov v uzle
  + M´´arny strom má najviac m potomkov a min m/2 potomkov
  + Štiepenie, vkladanie ... rovnaky link 43-50
* B+ strom
  + redundantná varianta Bstromu
  + vyvážený, listy na rovnakej úrovni
  + záznamy s dátami sú adresované iba z listov
  + všetky uzly sú aspoň napoly plné
  + vo vnútorných uzloch sa môžu opakovať tzv hraničné kľúče
  + vnútorné uzly rádu m až na koreň obsahujú m/2 ukazateľov a max m ukazateľov
  + listový uzol obsahuje ukazateľ na ďalší listový uzol, obsahuje aspoň m/2 ukazateľov a max m ukazateľov
  + 55-63
* Trie
  + E.Fredkin, 1960, information re-trie-val
  + datová štruktúra používaná pre reprezentáciu reťazcov nad danou abecedou
  + každý uzol obsahuje pole ukazateľov , každý jeden odpovedá znaku abecedy v ktorej sú kľúče vyjadrené
  + Všetci následníci uzlu majú spoločný prefix
  + koreň je prázdny reťazec
  + hodnoty sa asociujú s listami
  + není to m-árny vyhľadavací strom
  + použitie napr na rýchle vyhľadávanie vzorov, prefixov porovnávaním, ...
  + Uzol m-árneho trie je m prvkové pole => iný naázov pre tire je radix searching
  + vlastnosti – výška je daná dĺžkou najdlhšiho kľúča(slova), není nutné strom vyvažovať
  + príklady 72-113

# Hašovanie na vonkajších pamätiach

* Hašovacia funkcia m=h(k) – funkcia rieši prístup k obsahovo špecifikovateľným objektom so zložitosťou O(1)
* H = generátor rovnomerne rozložených hodnôt
* K – vyhľadavací kľúč
* M – adresa lokality kľúčom špecifikovaného objektu v pamäti
* Funkcia h by mala byť rýchlo a ľahko vypočítateľná
* K má byť numerické
* Zloženie typickej haš.funkcie :
  + 1. Generovanie kódu kľúča z hodnoty kľúča záznamu
    - Tj preklad hodnoty kľúča záznamu na celočíselnú hodnotu
  + 2. Kompresné zobrazenie kódu kľúča na adresu umiestnenia
    - Vždy nejaká forma zobrazenia celých čísel z jedného intervalu do iného
* Kolízia?
  + ak sa dva rôzne kľúče zobrazia na rovnakú adresu
  + redukcia kolízií
    - Lepšia hašovacia funkcia
    - Rozptyľovať do väčšieho priestoru
    - Do jedneho miesta umiestňovať viac objektov (kapsy, reťazené záznamy)
* Ak je hašovacia funkcia h prostá, hašovanie nazývame perfektné
* Ak je h prostá a priestor cieľových adreis má rozsah rovný počtu hodnôt kľúča, hašovanie voláme minimálne perfektné hašovanie
* Požadované vlastnosti hašovacej funkcie
  + deterministická = jej hodnoty závisia len na hodnotách kľúča
  + rychlá = niekoľko jednoduchých inštrukcií
  + každá jej hodnota závisí na hodnotách všetkých bitov kľúča alebo aspoň väčšiny
* Ideálna hašovacia funkcia
  + je prefektná ak každému miestu prideľuje 1 kľúč
  + je perfektne minimálna ak každému miestu prideľuje 1 kľúč a to v priestore ktorý obsahujepráve toľko miesta koľko je kľúčov
  + je náhodná
  + generuje funkčné hodnoty s uniformným rozložením
  + perfektné haš funkcie nemajú pri práci so súbprmi uplatnenie
* Najčastejšie používané funkcie
  + Modulo m=k mod M
    - Funkcia dáva zlé výsledky ak M nie je prvočíslo
    - Najčastejšie M je najmenšie prvočíslo väčšie než M
  + modifikácia modula konštantami napr m=(a \* k + b) mod M
  + mid square
    - Artimetika s veľkou presnosťou
    - K = 246-158-34 , 158^2 = 2-496-4, h(k) = 496
  + radix transformation
* Stratégia správy kolízií
  + open addressing resp. Closed hashing
    - Open v open addressing nam vraví, že cieľová adresa miesta v ktorom bude uloźený obejtk neni úplne určená hodnotou získanou hašovacou funkciou
    - Closed v closed hashing odkazuje na to, že nikdy neopustíme hašovaný priestor
  + closed addressing resp. Open hashing
    - Open v open hashing nám vraví, že žiadny z objektov v skutočnosti neni uložený v hašovanom priestore al v zozname ktorý je separovaný
    - Closed v closed addressing hovorí, že cieľová adresa miesta v ktorom bude uložený objekt je úplne určený hodnotou získanou v hašovacej funkcii
  + kolízie sa riešia vyhľadavaním alternativných miest v hašovanej štruktúre dokiaľ sa v nej nenájde cieľový objekt alebo prázdne miesto
    - kompresná funkcia vyráta z kódu kľúča „výchozí adresu“
    - v prípade kolízie sa pokračuje v hľadaní poč´´inajúc výchozí adresou špecifickým algoritmom
    - nikdy sa neopust´hašovaná štruktúra
  + lineárne riešenie kolizí
    - Pri vkladnaí sa použije prvé nasledujuce nájdené prázdne miesto
    - Pri hľadaní sa hľadá počínajúc výchozí adrsou prvé miesto obsahujúce hodnotu kľúča
    - Pri rušei sa nájde miesto odpovedajúce ruśenému záznamu a nahradí sa príznakom prázdnosti ktorý vraví že pri ďalšom hľadaní nech sa toto miesto preskakuje
  + kapsy
    - Jednotka pamäti obsahujúca jedne alebo viac objektov na vonkajšej pamäti je kapsou obvykle fyz.stránka
    - Umiestňujú sa do nej všetky kolizné prípady
    - Keď sa kapacita vyčerpá lze naviazať prietokovú kapsu
* Dynamické hašovanie
  + rozširiteľné hašovanie
    - Počet kapes pri určitom rozsahu kľúča sa môže meniť podľa okamžitych požiadaviek
    - Kapsy sa neadresujú priamo ale sú reprezentované poľom ukazateľov kapes – adresár kapes
    - Niesú nutné kapsy pre prietoky
    - Často kombinuje tries a hašovanie
    - Pre rychlý prístup k kapsám lze použiť ries
    - Klady – stabilný výkon, adresár sa vojde do RAM, malá priestorová réžia
    - Zápory – zmena rozmeru kapes vyvolá rozsiahle korekcie
  + lineárne hašovanie
    - Kapacita kapies je daná kapacitou bloku vonkajšej pamäti
    - Nepožaduje sa samostatná adresárová štruktúra
    - Začína sa s istým počtom kapes a keď sa v niektor prekročí faktor naplenneia, štiepi sa prvá kapsa. Táto sa umiestni na koniec súboru ako kopsa M a počet kapies je potom M+1
    - Záznamy zo štiepenej kapsy sa rozmiestnia medzi pôvodnu a novú
    - Nemusí sa vytvárať adresár kapes, adresový priestor sa rozširuje štiepením pri každom prietoku

PV062-organizacia-suborov-poznamky-otazky-4.6.2012.pdf

Vyborny document urcite ho prejdite

Nieco co je tam som ani nenasla v prezentaciach (napr. Fagin)

Ak si budete pozerat nejake stare pisomy z FI MUNY, taka k natrafite na nieco co tu nie je, chodte do slideov, snazila som sad at vsetko podla mna podstatne no moze sa stat ze Staudek povazuje nieco ine za podstatne

Brace yourselves, skuskove is coming