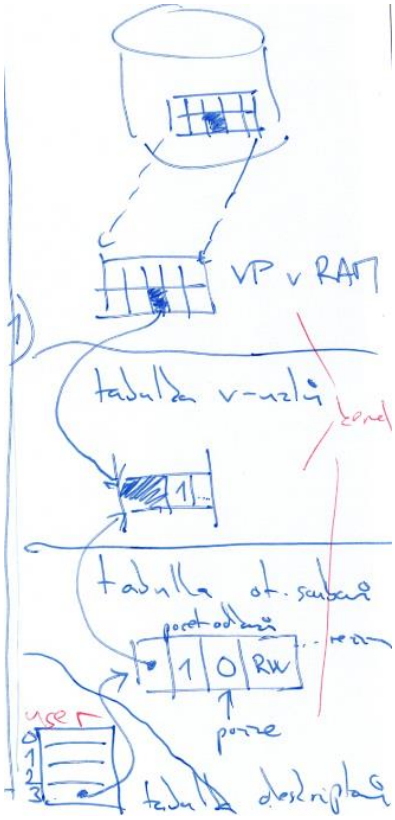
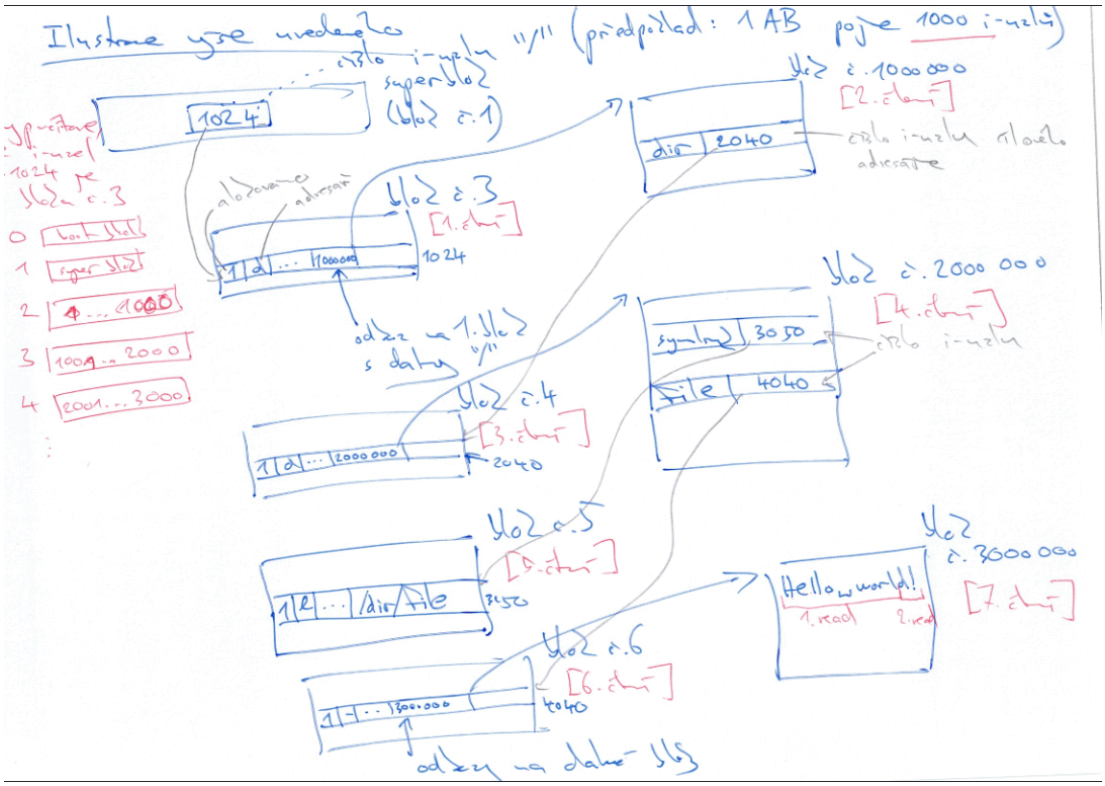
IOS - Cvičení - souborové systémy

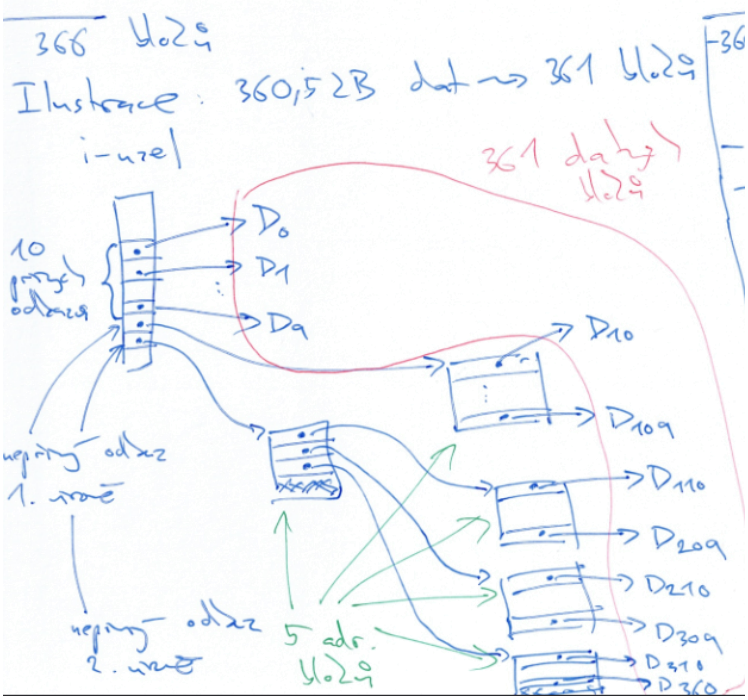


- **1.** Popište stručně typické kroky, kterými prochází voláni open() v UNIXU
  - open(„/bin/cat“, …)
  - Vyhodnotí se cesta k otevíranému souboru, ověří se přístupová práva, existence souboru, načte blok s **i-uzlem** souboru do vyrovnávací paměti (v RAM)
  - (V-uzel -> instance i-uzlu v ramce)
  - V tabulce v-uzlů si alokuju novou položku, do ní se načte příslušný i-uzel a doplní se dalšími daty, např. počet odkazů
  - V **tabulce otevření souborů** se **alokuje** nová položka, naplní **se odkazem na v-uzel a dalšími daty: počet odkazů, pozice v souboru (default 0), režim otevření, …**
  - V tabulce popisovačů souborů (**deskriptorů**) **se alokuje nejmenší volná položka** a naplní se odkazem do tabulky otevření souborů
  - Vrátí se **index alokované položky** v tabulce deskriptorů, nebo **-1 při chybě**

- **2.** Předpokládejte klasický UNIX-ový souborový systém FS, jaký je maximální počet čtení bloků z disku u sekvence operací:
  - $h = open(/\mathrm{dir/symlink}, \ldots);$
  - $read(\mathrm{h}, \mathrm{buf}, 10);$
  - $read(\mathrm{h}, \mathrm{buf}, 10);$
  - Na začátku jsou všechny vyrovnávací paměti prázdné (jsme v situaci tesně po připojení FS-mount), ale následně budou využity všechny běžné typy VP.
  - Soubor /dir/symlink je tzv. rychlý symlink odkazující na soubor /dir/file, což je obyčejný soubor o délce >20B
  - Všechny adresáře zabírají 1 blok. Všechny soubory existují, nedojde k přepnutí kontextu…

  - 7 čtení
    - **Na začátku víme jen číslo i-uzlu „/“. To je možné čistě aritmeticky převést na číslo loku na disku a načíst blok s i-uzlem „/“**
    - **Načte se blok s obsahem „/“ a nalezne se dvojice (dir, #i-uzel dir)**
    - **Načte se alokační blok s i-uzlem dir**
    - **Načte se blok s obsahem adresáře „/dir“**
    - **V datech „/dir“ je nalezena dvojice (symlink, #i-uzel symlinku), číslo i-uzlu se převede na číslo bloku s i-uzlem ten se načte**
    - Protože „/dir/symlink“ je rychlý symlink, přímo v i-uzlu je uvedená cesta „/dir/file“
    - Cesta „/dir“ je již zpracovaná a uložená v d-entry cache a obsah „/dir“ je též ve VP, postačí tedy v něm vyhledat dvojici (file, #i-uzel file)
    - **Číslo i-uzlu file se převede na číslo bloku a načte se blok s i-uzlem „/dir/file“**
    - **1. operace read načte 1. blok dat souboru „/dir/file“ do VP a vrátí 10B**
    - 2. operace read načte dalších 10B z VP bez nutnosti dalšího čtení z disku

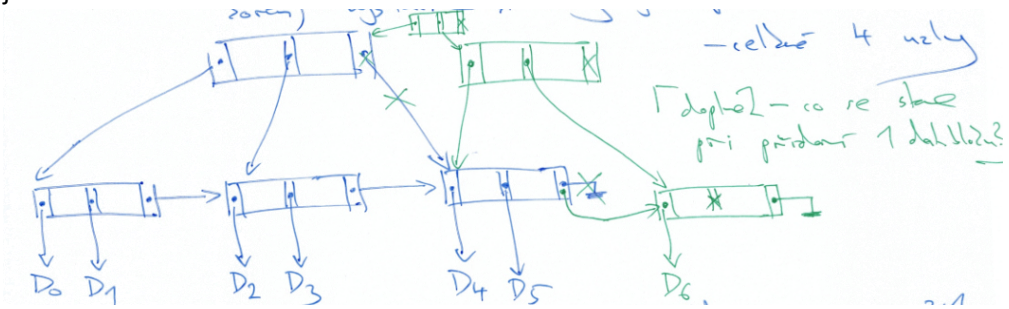
- 

- **3.** Uvažujme soubor v klasickém UNIXovém FS o velikosti 360,5kB, bloky o velikosti 1000B, odkazy mají 10B. Kolik zabírá soubor bloků včetně metadat (bez i-uzel -> jelikož jsou alokované předem, nezabírá víc místa ho mít/nemít, nepočítají se)
  - **366 bloků**
  - 

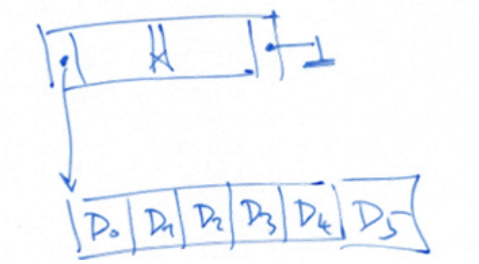
  - **360,5kb dat = 361 datových bloků**
  - (alokuje se po blocích)
  - **10 datových bloků je odkazováno přímo z i-uzlu**
  - **Dalších 100 (1000/10) datových bloků je odkazováno z pomocného adresového bloku přímých odkazů z i-uzlu**

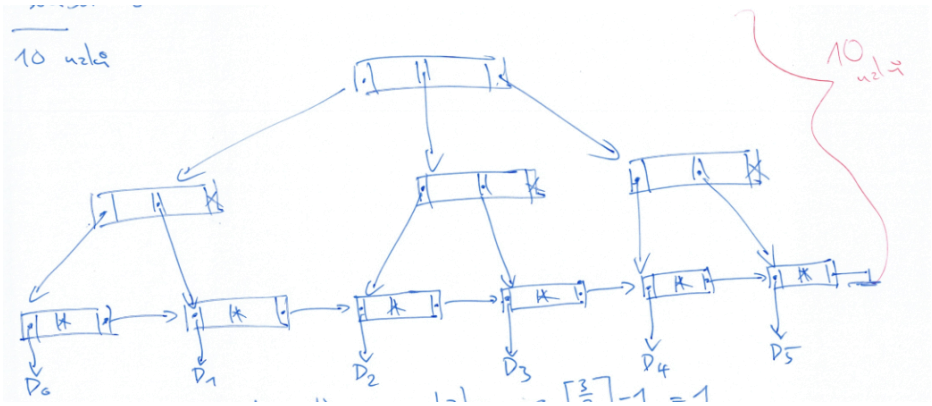
- **Zbývajících 251 datových bloků je odkazováno z dalších 3 bloků přímých odkazů, jež jsou odkazovány z pomocného bloku nepřímých odkazů 1. úrovně, který je dostupný z i-uzlu**
- **361+1+3+1=366**

- V klasickém UNIXovém systému:
  - 10 přímých odkazů na datové bloky (D0 – D9)
  - Nepřímé odkazy 1. úrovně – odkaz na blok který má přímé odkazy (tam je jich 100, jelikož velikost odkazu je 10B a blok má 1000B)
  - 1. Nepřímý odkaz (1. úrovně) – blok s odkazy D10-D109
  - 2. Nepřímý odkaz (2. úrovně) – ukazuje na blok plných nepřímých odkazů 1. úrovně
    - Položka 1 v bloku – D110-D209
    - Položka 2 v bloku – D210-D309
    - Položka 3 v bloku – D309-D360
    - … konec dat

- **4.** Kolik uzlů bude minimálně mít B+ strom stupně 3 (3 odkazy v uzlu) popisující soubor o 6 alokačních blocích, nejsou-li použitý extenty (neboli 1AB=1 extent)
  - **4. uzly**
  - Na listové úrovni lze z 1 listu odkázat max 2 datové bloky protože 1 odkaz potřebujeme pro vytvoření seznamu listů
  - Potřebujeme tedy 3 listové uzly ($\left\lceil \frac{6}{3-1} \right\rceil$)
  - Do kořene lze umístit až 3 odkazy (nejedná se o sólo kořen – kořen je jediný uzel) – vystavuje si tedy již jen s kořenem
  - 

  - **Pozor, na uzly pod kořenem není možné rozdělení na např. 3+1 odkazy limit (dolní) zaplnění je** $\left\lceil \frac{počet\_odkazů}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$
  - **U listů to je** $\left\lceil \frac{počet\_odkazů}{2} \right\rceil - 1 = \left\lceil \frac{3}{2} \right\rceil - 1 = 1$

- **4b.** Kolik uzlů by mohlo stačit v minimálním případě při použití extentů (extenty neomezené velikosti)
  - **Stačil by 1**
  - Všechny datové bloky mohou být v 1 extentu a bude tedy stačit 1 odkaz z kořene -> 1 uzel
  - 

- **4c.** Kolik uzlů B+ stromu může být použito v nejhorším případě pro soubor o velikosti 6 bloků
  - **10 uzlů**
  - 

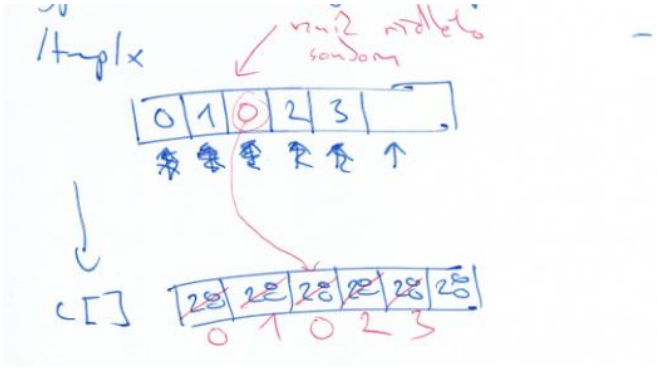
  - Minimální počet odkazů v listu je $\left\lceil \frac{3}{2} \right\rceil - 1 = 1$
  - Můžeme tedy využít 6 listů
  - Vnitřní uzly musí mít ale aspoň $\left\lceil \frac{3}{2} \right\rceil = 2$ odkazy
  - Vytvoříme tedy 3 uzly nad listy a pak již jen kořen (kořen má minimum 1)
  - Celkově tedy 10 uzlů

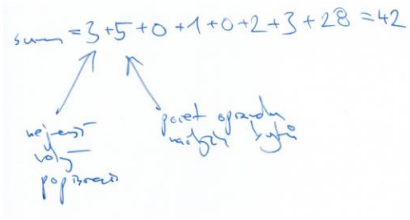
5, (5 bodů) Jaký součet vytiskne níže uvedený program, předpokládáme-li, že je spuštěn standardním způsobem z příkazové řádky a nenastane chyba při vytváření ani použití souboru /tmp/x? Výsledek pečlivě zdůvodněte. (limity: 1 číslo [bez vysvětlení za 0b.] a cca 5 rozvitých vět)

```
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/stat.h>

int main() {
        int fd, sum, i;
        char a[] = { 0, 1 };
        char b[] = { 2, 3 };
        char c[] = { 28, 28, 28, 28, 28, 28 };

        // Vytváříme soubor s právy vlastníka pro čtení i zápis a
        // otevíráme ho pro zápis. Existuje-li již soubor, je zkrácen
        // na soubor prázdný.
        fd = open("/tmp/x",O_WRONLY|O_CREAT|O_TRUNC,S_IRUSR|S_IWUSR);
        write(fd,a,2);
        lseek(fd,1,SEEK_CUR);
        write(fd,b,2);
        close(fd);
        sum = fd = open("/tmp/x",O_RDONLY); // Inicializace sum!
        sum += read(fd,c,6);                // Přičtení k sum!
        close(fd);
        for (i=0; i<6; i++) sum += c[i];    // Přičítání k sum!
        printf("the sum is %d\n",sum);
        return 0;
}
```

-
  - Finální číslo = 42
  - 

  - 

  - **Počáteční suma je 3**
    - **3 je nejmenší volné číslo popisovače**
    - **0 – stdin, 1 – stdout, 2 – stderr**
  - **Read chce načíst 6 bytů ale soubor má jen 5, tedy read vrátí 5**
  - **Read přepíše prvních 5 prvků c[] hodnotami:**
    - **0, 1 – z původního a[]**
    - **0 – důsledek řídkého souboru**
    - **2,3 – původní b[]**
  - **Poslední prvek c[] zůstane beze změny**

- **6.** Navrhněte fragment extentového stromu FS ext4, který bude popisovat soubor tvořený alespoň 2 extenty (1. o velikosti 2 bloky, 2. o velikosti 4 bloky), přičemž bude mít alespoň 1 nelistový uzel, který současně není kořenovým a alespoň 1 listový uzel. Veškeré adresování je na úrovni AB.
    - **V i-uzlu máme kořenový uzel**
    -