**Popište stručně typické kroky kterými prochází volaní open() v UNIXU.**

1. Vyhodnotí se cesta k otevíranému souboru, ověří se přístupová práva, existence souboru, načte blok s I-uzlem souboru, uloží ho do vyrovnávací paměti
2. V tabulce V-uzlů se alokuje nová položka, do ní se načte příslušný I-uzel a doplní se dalšími daty
3. V tabulce otevření souborů se alokuje nová položka, naplní se odkazem na V-uzel a dalšími daty: počet odkazů, pozice souboru, režim otevření,…
4. V tabulce popisovačů souborů (deskriptorů) se alokuje nejmenší možná položka a naplní se odkazem do tabulky otevření souborů
5. Vrátí se index alokované položky v tabulce deskriptorů, nebo -1 při chybě



**Předpokládejme klasický UNIXový souborový systém FS. Jaký je maximální počet čtení bloků z disku u sekvence operací:**

**h = open(„/dir/symlink“);**

**read(h,buffer,10);**

**read(h,buffer,10);**

**Na začátku jsou všechny vyrovnávací paměti prázdné, ale následně budou maximálně využity všechny běžné typy vyrovnávacích pamětí. Soubor /dir/symlink je tzv. rychlý symlink odkazující na soubor /dir/file, což je obyčejný soubor o délce větší než 20B. Všechny adresáře zabírají 1 blok. Všechny soubory existují.**

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



**Uvažujme soubor v klasickém UNIXovém FS o velikosti 360,5 kB. Bloky o velikosti 1000B, odkazy mají 10B. Kolik zabírá soubor bloků včetně metadat?**

360,5 kB dat vyžaduje 361 alokačních bloků (kolik 1kB bloků potřebuju na pokrytí 360,5 kB dat)

Prvních 10 dat. bloků je odkazováno z i-uzlu (V klasickém UNIX FS)

Dalších 100 (1000/10) datových bloků je odkazováno z adresového bloku přímých odkazů z i-uzlu

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

**Výpočet:** (Dn = Datový blok s indexem n)

První 10 je odkazováno z I-uzlu -> 361 – 10 = 351 (D0 – D9 mám hotovo)

Kolik dat. bloků bude mít blok přímých odkazů? 1000 / 10 = 100

Na to potřebuji nepřímý odkaz 1. úrovně, který má v sobě blok přímých odkazů (D10 – D109 mám hotovo) 351 – 100 = 251

Následuje nepřímý odkaz 2. úrovně (má v sobě blok nepřímých odkazů odkazující na bloky přímých odkazů)

1 nepřímý odkaz – 1 blok přímých odkazů (D110 – D209) 251 – 100 = 151

2 nepřímý odkaz – 2 blok přímých odkazů (D210 – D309) 151 – 100 = 51

3 nepřímý odkaz – 3 blok přímých odkazů (D310 – D360) 51 – 51 = 0

**Součet**: 361 (počet alokačních bloků)

+ 1 (blok přímých odkazů D10 – D109)

+ 1 (blok nepřímých odkazů)

+ 1 (blok přímých odkazů D110 – D209)

+ 1 (blok přímých odkazů D210 – D309)

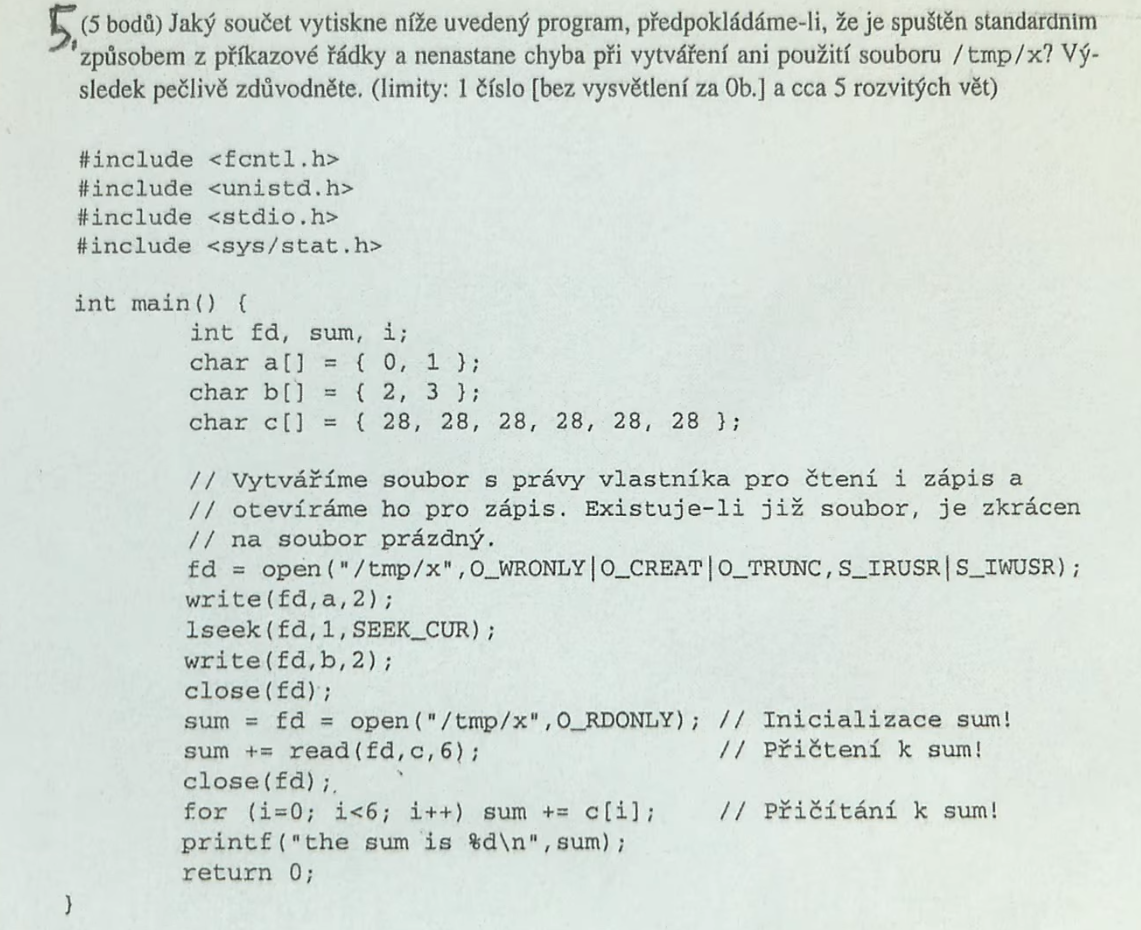
+ 1 (blok přímých odkazů D110 – D209) = **366**

**Kolik uzlů bude minimálně mít B+ strom stupně 3 (3 odkazy v uzlu) popisující soubor o 6 alokačních blocích, nejsou-li použity extenty (extent má velikost 1 bloku)**

1 uzel má 3 odkazy -> 2 odkazy na alokační bloky, 1 odkaz na další uzel

1. Uzel: Odkaz na D0 , odkaz na D1, odkaz na uzel 2
2. Uzel: Odkaz na D2 , odkaz na D3, odkaz na uzel 3
3. Uzel: Odkaz na D4 , odkaz na D5, prázdný odkaz
4. Uzel: Odkaz na uzel 1, odkaz na uzel 2, odkaz na uzel 3

Odpověď: Bude mít minimálně **4** uzly.



Otevře se soubor

Zapíšeme do souboru 0, 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | Jsem tady |  |  |  |  |  |  |  |

Lseek – posunu se o jedno doprava

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 |  | Jsem tady |  |  |  |  |  |  |

Zapíšeme do souboru 2, 3

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 |  | 2 | 3 | Jsem tady |  |  |  |  |

Zavře se soubor, na prázdná místa přijde 0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 2 | 3 |  |  |  |  |  |

Otevřu soubor a do sum se načte 3 (musím si pamatovat)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0  1. Bajt | 1  2. Bajt | 0  3. Bajt | 2  4.Bajt | 3  5.Bajt |  |  |  |  |  |

K sum se přičte počet zaplněných bajtů, tedy 5 (maximálně 6, protože sum+= read(fd, c, 6))

sum = 3 + 5 = 8

Zárověň se do c načtou hodnoty na daných bajtech

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 2 | 3 | 28 |

Zavře se soubor

Přes cyklus projdeme pole c a hodnoty na daných indexech přičteme k sum

sum = 8 + 0 + 1 + 0 + 2 + 3 + 28 = **42**

Program vypíše, že sum = 42

**Navrhněte fragment extentového stromu souborového systému ext4, který bude popisovat soubor tvořený minimálně 2 extenty, první o velikosti 2 bloky, druhý o velikosti 4 bloky, přičemž bude mít alespoň jeden vnitřní nelistový uzel, který současně není kořenový a alespoň jeden listový uzel. Veškeré adresování je na úrovní alokačních bloků.**

**Obsah obrázku diagram, text, Paralelní, Plán

Obsah vygenerovaný umělou inteligencí může být nesprávný.**

**Vysvětlivky:**

LB – Logický blok

FB – Fyzický blok

PB – Počet bloků