OS MMXX

MIT ;)

https://pdos.csail.mit.edu/6.828/2020

Suborovy system

Preco potrebujeme FS?

- Perzistencia medzi restartom systemu
- Hierarchia, organizacia, pomenovanie dat
- Zdielanie (medzi aplikaciami a pouzivatelmi)

Co je na navrhu FS zaujimave

- Obnova po zlyhani (*crash recovery*)
- Vykon (konkurentne vykonavanie)
- Zdielanie
- Bezpecnost
- Abstrakcia jednotneho pristupu
  - /proc, /sys, …
  - rury, sokety
  - zariadenia

Priklad UNIX-like systemov

- fd = open(“x/y.txt”, …)
- write(fd, “abc”, 3)
- link(“x/y.txt”, “x/z.txt”)
- write(fd, “def”, 3)
- close(fd)

- Subor y.txt aj z.txt (ten isty subor, dve rozne cesty k jeho obsahu) obsahuje “abcdef”

UNIX FS API

- Objekty
- Obsah (data)
- Pomenovania
- Organizacia
- Synchronizacia

UNIX FS API

- Objekty: subory
- Obsah (data): pole bajtov
- Pomenovanie: ludsky citatelne
- Organizacia: hierarchia mien
- Synchronizacia: nie je

Niektore dosledky API

Niektore dosledky API

- fd odkazuje na nieco, co sa zachovava, aj ked
  - sa meno suboru zmeni
  - sa meno zmaze (pokym ostava subor otvoreny)

Niektore dosledky API

- fd odkazuje na nieco, co sa zachovava, aj ked
  - sa meno suboru zmeni
  - sa meno zmaze (pokym ostava subor otvoreny)
- Subor moze mat viacero liniek
  - T.j. data su dostupne pomocou viacerych ciest
  - Ziadna cesta nema prioritu

Niektore dosledky API

- fd odkazuje na nieco, co sa zachovava, aj ked
  - sa meno suboru zmeni
  - sa meno zmaze (pokym ostava subor otvoreny)
- Subor moze mat viacero liniek
  - T.j. data su dostupne pomocou viacerych ciest
  - Ziadna cesta nema prioritu
- Takze info o subore musi byt niekde inde ako v adresari

Niektore dosledky API

- FS uchovava info o subore v tzv. i-uzle (*inode*) na disku
- FS odkazuje na i-uzol pomocou i-cisla (*inumber*) – ide o nieco podobne ako FD

Niektore dosledky API

- FS uchovava info o subore v tzv. i-uzle (*inode*) na disku
- FS odkazuje na i-uzol pomocou i-cisla (*inumber*) – ide o nieco podobne ako FD

- i-uzol musi obsahovat pocitadlo referencii (liniek)
- i-uzol musi udrziavat pocet otvorenych FD
- Uvolnenie i-uzla je odlozene, pokym pocet liniek a pocet otvorenych FD neklesne na 0

Disk

- Udaje sa uchovavaju na perzistentnom mediu
- Najpouzivanejsie media
  - HDD (velka kapacita, ale pomale; lacne)
  - SSD (mensia kapacita, ale rychle; drahe)

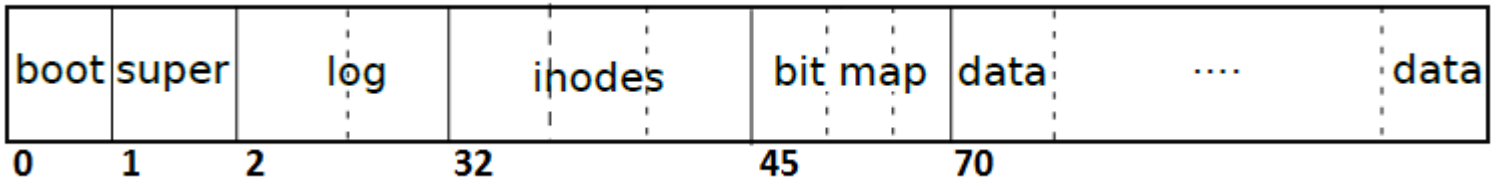
Disk

- Udaje sa uchovavaju na perzistentnom mediu
- Najpouzivanejsie media
  - HDD (velka kapacita, ale pomale; lacne)
  - SSD (mensia kapacita, ale rychle; drahe)

- Najmensia adresovatelna jednotka pri praci s diskom je 1 sektor = 512B (vid ramec v RAM)
- OS pracuje s blokmi (nasobok sektora)
  - xv6 pouziva bloky o velkosti 2 sektorov (1kB)

Struktura disku xv6

- Blok 0: nepouzity
- Blok 1: super blok (velkost fs, pocet i-uzlov)
- Blok 2: zaciatok blokov pre transakcie FS
- Blok 32: pole i-uzlov
- Blok 45: blok bitmapy vyuzitia blokov (0=volny blok, 1=pouzity blok)
- Blok 70: zaciatok blokov pre obsah suborov a adresarov



Struktura disku xv6

- Kto vytvara tuto strukturu?
- Program mkfs (priecinok mkfs)

Struktura disku xv6

- Kto vytvara tuto strukturu?
- Program mkfs (priecinok mkfs)

- Co su metadata?
- Vsetko na disku okrem samotneho obsahu suborov
  - Superblok
  - i-uzly
  - Bitmapy
  - Obsah adresarov

I-uzol na disku

- Co obsahuje i-uzol na disku?
  - Typ (volny, subor, adresar, …) (type)
  - Pocet odkazov na subor z adresarovej struktury (nlink)
  - Velkost suboru (size)
  - Odkazy na data suboru (addrs[12+1])
- Obrazok priamych, nepriamych, dvojito nepriamych blokov

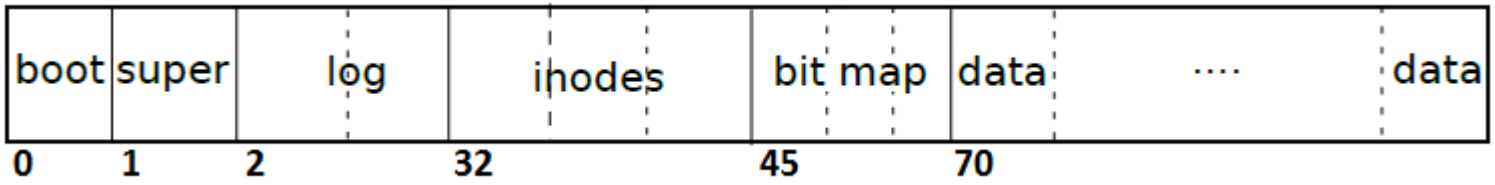
Vypocet logickeho bloku

- Ako najst adresu bloku disku, v ktorom sa nachadza konkretny bajt suboru?
- Priklad: 8000-ci bajt suboru
  - Logicky blok = 8000/BSIZE = 8000/1024 = 7
  - Polozka v poli adries blokov suboru (addr) s ind 7
- Priklad: 20134-ty bajt suboru
  - Logicky blok = 20134/BSIZE = 19
  - Kde sa nachadza adresa logickeho bloku suboru c.19?

I-uzol xv6 na disku

- Ako najst i-uzol na disku?

- Kazdy i-uzol ma svoje cislo *inum* (index do pola i-uzlov na disku)
- i-uzol ma velkost 64B
- Adresa i-uzla na disku: 32*BSIZE + 64*inum



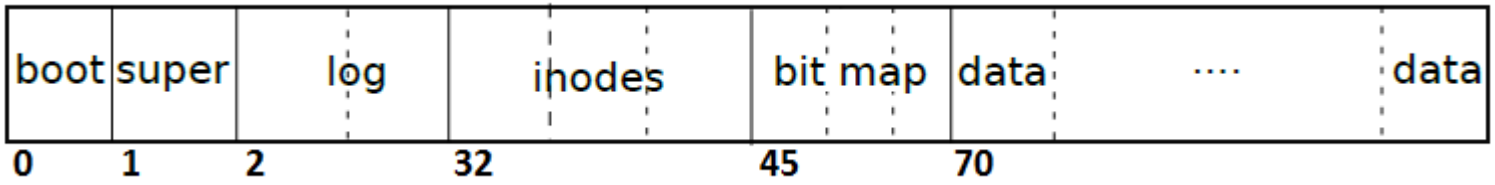
I-uzol xv6 na disku

- Struktura i-uzla v xv6

- type
- nlink
- size
- addrs[NDIRECT+1]

Obmedzenia FS v xv6

- Maximalny pocet suborov/adresarov
- Maximalna velkost suboru/adresara
- Maximalna velkost suboroveho systemu (disku)



Obsah adresara

- Adresar je tiez len subor (pole bajtov)
  - Ale toto pole bajtov dokaze interpretovat jadro OS
  - K tomuto polu bajtov nema priamy pristup pouzivatel (iba pomocou systemovych volani na pracu s FS)

Obsah adresara

- Adresar je tiez len subor (pole bajtov)
  - Ale toto pole bajtov dokaze interpretovat jadro OS
  - K tomuto polu bajtov nema priamy pristup pouzivatel (iba pomocou systemovych volani na pracu s FS)
- Subor adresara sa v xv6 interpretuje ako pole struktur *dirent* (kernel/fs.h)
  - Velkost 16B
    - Meno polozky iba 14B
    - 2B pre cislo i-uzla, ktory popisuje obsah suboru
  - Struktura je volna ak je inum rovne 0

Spracovanie cesty v xv6

- Tzv. *pathname lookup* (napr. “/x/y”)

Spracovanie cesty v xv6

- Tzv. *pathname lookup* (napr. “/x/y”)

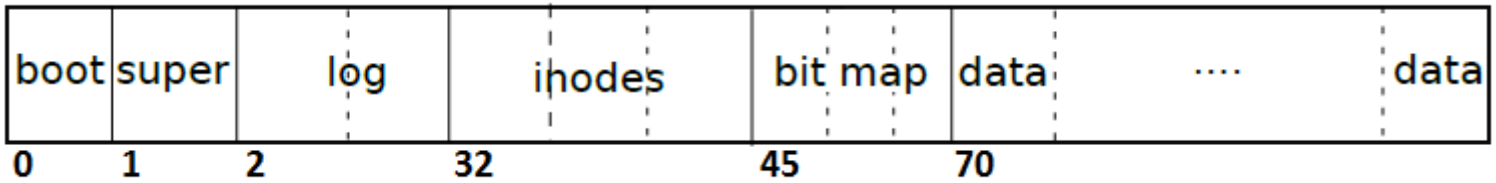
- “/” root → natvrdo dane cislo root i-uzlu (1)
- V obsahu suboru, ktory popisuje i-uzol 1, hladaj *direntry* s menom “x”
- Ak najdes polozku “x”, v obsahu suboru, ktory popisuje i-uzol tejto polozky (v blokoch suboru “x”), hladaj direntry s menom “y”
- Ak najdes polozku “y”, nasiel si aj cislo i-uzla, ktory popisuje subor “y”

Ukazka xv6

- Zameriame sa na zapis blokov na disk

- Vytvorenie suboru
- Zapisanie do suboru
- Zmazanie suboru

- rm fs.img && make qemu



$ echo hi > x

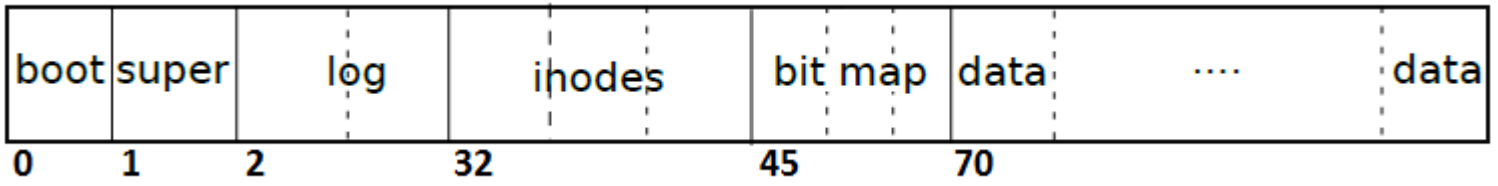
```
// create
bwrite: block 33 by ialloc()      // alloc inode in inode block 33
bwrite: block 33 by iupdate()     // update nlink in inode
bwrite: block 70 by writei()      // write direntry (‘x’ by dirlink())
bwrite: block 32 by iupdate()     // update size dir inode
bwrite: block 33 by iupdate()     // itrunc() new inode
// write
bwrite: block 45 by balloc()      // alloc a block in bitmap block 45
bwrite: block 644 by bzero()      // zero the allocated block
bwrite: block 644 by writei()     // write ‘hi’ to it
bwrite: block 33 by iupdate()     // update size in inode
// write
bwrite: block 644 by writei()     // write ‘\n’ to it
bwrite: block 33 by iupdate()     // update size in inode
```

Graf volani

sys_open	sysfile.c
create	sysfile.c
ialloc	fs.c
iupdate	fs.c
dirlink	fs.c
writei	fs.c
iupdate	fs.c
itrunc	sysfile.c
iupdate	fs.c

$ rm x

```
bwrite: block 70 by writei()      // from sys_unlink; directory content
bwrite: block 32 by iupdate()     // from writei of directory content
bwrite: block 33 by iupdate()     // from sys_unlink; link count of file
bwrite: block 45 by bfree()       // from itrunc; from iput
bwrite: block 33 by iupdate()     // from itrunc; zeroed length
bwrite: block 33 by iupdate()     // from iput; marked free
```



Graf volani

sys_unlink	sysfile.c
writei	fs.c
iupdate	fs.c
iunlockput	fs.c
iput	fs.c
itrunc	fs.c
bfree	fs.c
iupdate	fs.c
iupdate	fs.c

Domace citanie

Chapter 8

File System

xv6: a simple, Unix-like teaching operating system

Okrem kapitol a casti, ktore hovoria o logovani transakcii FS