Levytiedostojen käsittely

- Tietokantojen tietoja säilytetään yleensä apumuistissa, lähinnä levymuisteissa
- Apumuistiin tallentamisen merkittäviä etuja keskusmuistiin nähden ovat
  - tiedon säilyvyys (virtakatkon yli)
  - säilytyskapasiteetin edullisuus keskusmuistiin verrattuna, ja
  - suuremmat tallennusvolyymit – keskusmuistiakin mitataan nykyään jo gigatavuissa mutta tietokantoja jopa teratavuissa

Levytiedostojen käsittely

- Levymuistin merkittävin huono puoli keskusmuistiin nähden on tiedon hidas saantiaika
  - Keskusmuistissa saantiaika on kymmeniä nanosekunteja, luokkaa 10-50 ns
  - Levymuistista saantiaika on luokkaa 5-10 millisekuntia (ms) eli 5,000,000 - 10,000,000 ns
  - eli 100 000 – 1M hakua keskusmuistista yhtä levyhakua kohti

Levytiedostojen käsittely

- Ohjelmointikielissä levylle tallennettua tiedostoa käsitellään yleensä peräkkäisten tietueiden joukkona. Suorasaantitiedostossa (direct access file) tietue (record) voidaan osoittaa tiedoston sisäisen järjestysnumeron perusteella. Peräkkäissaantitiedostossa tietuetta ei voida hakea osoitteen perusteella vaan tiedostoa on käytävä läpi alusta alkaen kunnes haluttu tietue löytyy.

Levytiedostojen käsittely

- Tiedostorakennetasolla tiedosto voidaan ajatella tietueiden joukkona. Tyypillisiä tiedostoon kohdistuvia operaatioita ovat:
- **avaus** (open) – haetaan tiedoston liittyvät metatiedot (tiedostokuvaaja, file header) ja alustetaan tiedoston käsittelyyn tarvittavat tietorakenteet
- **sulkeminen** (close) – tiedoston käsittelyyn varatut resurssit vapautetaan
- **siirtyminen alkuun** (reset), vuorossaolevaa tietueen osoitin siirretään tiedoston alkuun.

Levytiedostojen käsittely

- ensimmäisen ehdon täyttävän tietueen haku (find) – ensimmäinen hakuehdon täyttävä tietue haetaan ohjelman työalueelle – eri tiedostorakenteet tukevat erilaisia hakuehtoja – yksinkertaisimmassa tapauksessa käytettävissä on vain haku tietueen järjestysnumeron (osoitteen) perusteella
- seuraavan hakuehdon täyttävän tietueen haku (find next)

Levytiedostojen käsittely

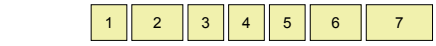
- tietueen poisto (delete) - tietue poistetaan tiedostosta
- tietueen muutos (modify) – jonkin tietoalkion arvon suhteen muuttunut tietue korvaa olemassaolevan tietueen
- tietueen lisäys (insert) – tiedostoon lisätään tietue
- ylläpito-operaatioiden tarjonta riippuu tiedostorakenteesta. Kaikki rakenteet eivät esimerkiksi tue poisto-operaatiota, vaan koko tiedosto poistettavaa tietuetta lukuun ottamatta on kirjoitettava uusiksi ja tällä tiedostolla sitten korvattava edellinen versio.

Levytiedostojen käsittely

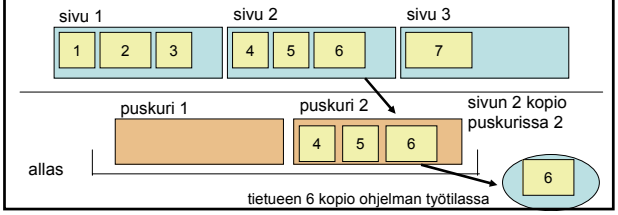
- Apumuistiin tallennettuja tietoja ei siirretä suoraan apumuistista ohjelman työtilaan vaan siirto tapahtuu puskureiden (buffer) kautta
- Kun ohjelma haluaa hakea tietyn tietueen on tietueen sisältävä sivu (page) haettava ensin johonkin puskuriin (buffer frame), jonka jälkeen tietue voidaan siirtää puskurista ohjelman työtilaan. Puskurien koon ja tiedoston sivukoon on vastattava toisiaan.
- Tiedoston käsittelyä varten on yleensä käytössä usean puskurin puskuriallas (buffer pool)

Levytiedostojen käsittely

Tiedostorakennetason näkemys: tiedostossa osoitettavissa olevia tietueita



Puskurinhallintatason näkemys: tietueet sijoitettu sivuille, sivuja käsitellään puskureiden kautta :



Levytiedostojen käsittely

Tietueen haku ohjelman käyttöön, karkea algoritmi:

```
tiedostorakenne.hae_tietue(x,muuttuja m)  {
  y= tiedostorakenne.päättele_sivu_tietueelle(x);
  p= puskurinhallinta.anna_sivu(y) {
       q= puskurinhallinta.missä_puskurissa_on_sivu(y);
       jos (q==null)  {     // ei ole missään
          q= puskurinhallinta.valitse_vapaa_puskuri;
          puskurinhallinta.lataa_sivu(puskuriin q sivu y);
          puskurinhallinta.naulitse(sivu y puskuriin q); }
       muuten  {
          puskurinhallinta.naulitse(sivu y puskuriin q);}
       palauta_osoitin_puskuriin q;
  }
  s= tiedostorakenne.päättele_sijainti(tietuelle x, puskurissa p);
  tiedostorakenne.siirrä(muutujaan m, puskurista p, kohdasta s);
}
```


Levytiedostojen käsittely

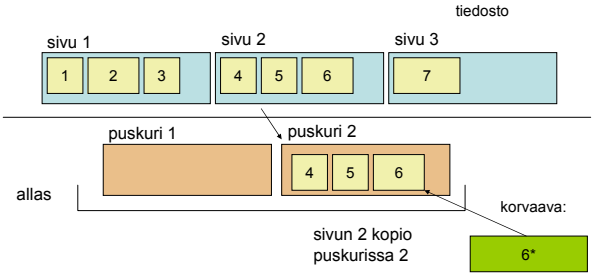
- Puskurinhallinnan on tiedettävä minkä sivun kopioita puskureissa on ja mitkä puskurit ovat vapaina tiedonsiirtoa varten.
- Naulitsemismenettelyllä (fix, pinning) pidetään kirjaa varatuista puskureista – naulitseminen kasvattaa puskuroidun sivun käyttäjälaskuria (pin counter). Kun sivun tarvitsija on käsitellyt sivun se ilmoittaa vapauttavansa puskurin (unfix), jolloin laskurin arvoa vähennetään. Kun yksikään prosessi ei enää tarvitse puskurissa olevaa sivua, on puskuri vapaa uudelleenkäyttöön.

Levytiedostojen käsittely

- Myös kirjoitukset levylle tapahtuvat puskureiden kautta. Uusi tai muuttunut tietue siirretään puskuriin. Puskuri merkitään muuttuneeksi (dirty). Kun prosessi on vapauttanut puskurin, puskurienhallinta siirtää puskurissa olevan sivun aikanaan levylle.
- Muutos voi kasvattaa sivun sisältöä niin, ettei se enää mahdu puskuriin. Tällöin otetaan tyypillisesti käyttöön ylivuotosivu, joka linkitetään alkuperäiseen sivuun.

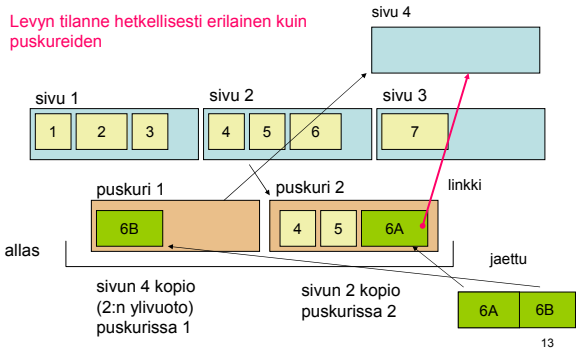
Levytiedostojen käsittely

Tietueen pituus kasvaa:



Levytiedostojen käsittely

Levyn tilanne hetkellisesti erilainen kuin puskureiden



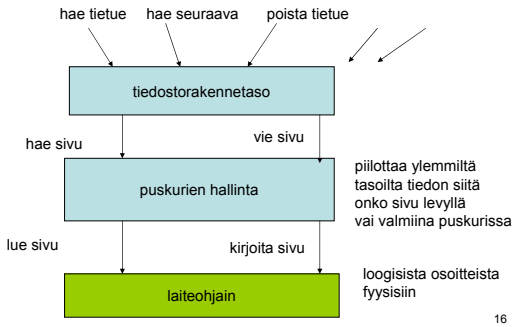
Levytiedostojen käsittely

- Jos puskurialtaat eivät ole prosessikohtaisia, voi usea prosessi saada edellä hahmotellussa puskurikäsittelyssä haltuunsa osoittimen samaan puskuriin ja voisi siis käsitellä puskurissa olevia tietoja. Tästä aiheutuvia samanaikaisuusongelmia hoidetaan salpojen (latch) avulla.
- salpoja on kahta tyyppiä
  - lukusalpoja (read latch) ja
  - kirjoitussalpoja (write latch) – estää muilta kaiken prosessoinnin
  - Vain yhdellä prosessilla voi olla kirjoitussalpa hallussaan

Levytiedostojen käsittely

- Sivulla voi
  - olla useita tietueita tai
  - olla vain osa tietueesta (tietue voi jakautua useille sivuille).
- Sivu on abstraktio fyysisen levyn **lohko** (block) käsitteestä. Joissakin järjestelmissä on mahdollista siirtää kerralla useampien lohkojen **ryppäitä** (cluster). Tällöin sivu vastaisi rypästä.
- Jos tietue jakautuu useille sivuille käytetään usein edellisen kuvan mallin mukaista ylivuotosivua

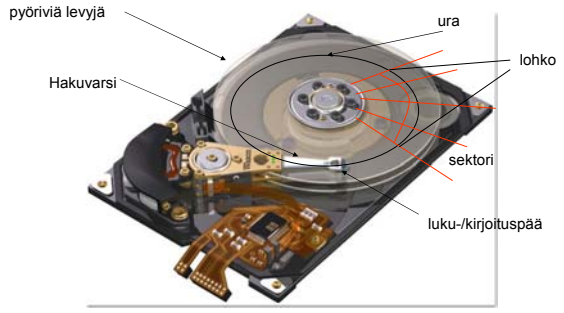
Levytiedostojen käsittely



Levytiedostojen käsittely

- Tiedostorakennetason käsittelyssä haku- operaation hakukriteerinä voi olla jonkin tietoalkion arvo.
- Puskuritason hakuoperaatiossa sivu haetaan loogisen sivuosoitteen perusteella
- Yhtä puskuria voi samanaikaisesti käsitellä vain yksi prosessori.
- Puskureiden ja apumuistin väliseen tiedonsiirtoon käytetään erillistä I/O- prosessoria. Täten yhdessä puskurissa olevaa tietoa pystytään käsittelemään samanaikaisesti kun toiseen puskuriin kohdistuu tiedonsiirtoa.

Levykön rakenne



Levyn rakenne

- Levykössä (disk drive) on useita samankeskisiä levyjä (disk)
- Levyissä on magneettinen pinta (disk surface) kummallakin puolella levyä
- Levyllä on osoitettavissa olevia uria (track), muutamasta sadasta muutamaan tuhanteen, päällekkäiset urat muodostavat sylinterin (cylinder)
- Urat voivat jakautuvat sektoreihin (sector)

Levyn rakenne

- Levyn formatoinnin yhteydessä levylle kiinnitetään urien jako lohkoihin (block). Lohkokoko on yleensä 512 tavun monikerta (kakkosen potenssi). Lohkot erotetaan toisistaan lohkovälillä, johon talletetaan lohkoon liittyvää kontrollitietoa.
- Lohko on pienin osoitettavissa ja siirrettävissä oleva kohde.
  - lohkon osoite muodostuu **levypinnan numerosta**, **uran numerosta** ja uran sisäisestä **lohkon numerosta**

Levyn rakenne

- Lohkon sisällön hakemiseksi
  - Siirretään luku/kirjoituspäät halutulle sylinterille (kaikki siirtyvät yhdessä)
  - Odotetaan, että haluttu lohko pyörähtää luku/kirjoituspään kohdalle (pyörimisnopeus luokkaa 100 kierrosta sekunnissa)
  - Aktivoidaan halutulla pinnalla oleva luku/kirjoituspää lukemaan dataa – yleensä vain yksi pää voi olla samanaikaisesti aktiivinen
  - I/O prosessori siirtää luetun datan hakupyynnön yhteydessä annettuun puskuriin

Levyn rakenne

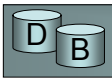
- Levyltä hakemiseen kuluva aika muodostuu
  - hakuvarren siirtoajasta eli kohdistusajasta (seek time) (sylinterille siirtyminen)
    - aika riippuu siitä millä sylinterillä päät ovat alunperin, lyhyt siirtymä vie vähemmän aikaa, 0 – levykohtainen maksimi. Usein ilmoitetaan vain keskimääräinen kohdistusaika esim. 6 ms
  - pyörähdysviiveestä (rotational delay)
    - riippuu alkuperäisestä kohdasta ja pyörimisnopeudesta.
    - jos levy pyörisi 100 kierrosta sekunnissa (6000 minuutissa) olisi keskimääräinen pyörähdysviive n. 5 ms (puolet yhden kierroksen ajasta).

Levyn rakenne

  - siirtoajasta (block transfer time)
    - riippuu uran kapasiteetista, pyörimisnopeudesta ja lohkon koosta
    - esimerkiksi jos
      - *pyörimisnopeus olisi 100 kierrosta sekunnissa*
      - *levyjä olisi 6 (12 pintaa) sylintereitä 8000 ja lohkokoko 1K tavua sekä uralla 500 lohkoa*
      - *niin levyn kapasiteetti olisi 1KB*500*96000 eli n 48 GB*
      - *niin lohkon siirtoaika olisi (1/500)*10 ms = 20000ns (pyörähdyksen aikana siirrettävä koko urallinen)*
      - *ja siirtonopeus 500KB/10ms= 50 KB/ms eli 50 MB/s*

Levyn rakenne

- Mitä lähempänä (oikeaan suuntaan) nykyistä kohtaa seuraava haettava jakso on sitä nopeampaa on haku, nopeasta hitaaseen
  - seuraavana samalla uralla
  - samalla sylinterillä 'seuraavana' – ei pyörähdysviivettä
  - jossain samalla uralla tai sylinterillä
  - läheisellä sylinterillä
  - kaukaisella sylinterillä



Levyn rakenne

- Peräkkäisten saman uran ja sylinterin jaksojen siirtäminen on tehokkainta koska tällöin ei tule hakuvarren siirtoa eikä pyörähdysviivettä.
- Levyjen siirtonopeudet ilmoitetaan usein bitteinä sekunnissa. Voidaan antaa joko sisäinen siirtonopeus ja tehollinen siirtonopeus (formatoituna)