Tiedostojen 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

1

Tiedostojen 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

2

Tiedostojen käsittely

- Tiedostoja on eri luonteisia:
  - Tiedostoa voidaan käsitellä yhtenä kokonaisuutena (esimerkiksi kuvatiedostot), jolloin tiedosto ladataan käsittelyä varten kokonaisuudessaan keskusmuistiin ja samoin tallennetaan kokonaisuutena
  - Tietokantojen yhteydessä tiedostot hahmotetaan yleensä tietueiden joukkoina (file of records). Käsiteltävät kokonaisuudet voivat tällöin olla yksittäisiä tietueita tai useiden tietueiden joukkoja.
- Kummatkin abstraktiot ovat tarjolla useimmissa ohjelmointikielissä. Tällä kurssilla käsitellään tietueiden joukko abstraktion mukaisia tiedostoja.

3

Tiedostojen käsittely

- Ohjelmointikielissä tietueiden joukkoa voidaan käsitellä joko
  - **peräkkäissaantirakenteena** (sequential access) tai
  - **suorasaantirakenteena** (direct access)
- Suorasaantitiedostossa (direct access file) tietue (record) voidaan osoittaa haun (tai tallennuksen) kohteeksi tiedoston sisäisen tietuetunnuksen (record identifier, rid) (=osoitteen) perusteella. Tällainen tunnus voi olla esimerkiksi tietueen järjestysnumero tai tavuosoite tiedoston alusta laskettuna.
- Peräkkäissaantitiedostossa tietuetta ei voida hakea osoitteen perusteella vaan tiedostoa on käytävä läpi tietue kerrallaan alusta alkaen kunnes haluttu tietue löytyy.

4

Tiedostojen käsittely

- 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), vuorossa olevan tietueen osoitin siirretään tiedoston alkuun.

5

Tiedostojen käsittely

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

6

Tiedostojen käsittely

- tietueen poisto (delete) - tietue poistetaan tiedostosta
- tietueen muutos (modify) – jonkin tietoalkion arvon suhteen muuttunut tietue korvaa olemassa olevan tietueen
- tietueen lisäys (insert) – tiedostoon lisätään tietue

- Tiedoston 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 on sitten korvattava edellinen versio.

7

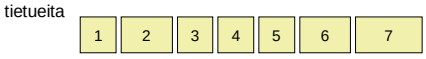
Tiedostojen käsittely

- 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 puskurikehykseen (buffer frame), jonka jälkeen tietue voidaan siirtää ohjelman työtilaan. Puskurikehysten koon ja tiedoston sivukoon on vastattava toisiaan.
- Tiedoston käsittelyä varten on yleensä käytössä usean puskurikehyksen puskuriallas (buffer pool)

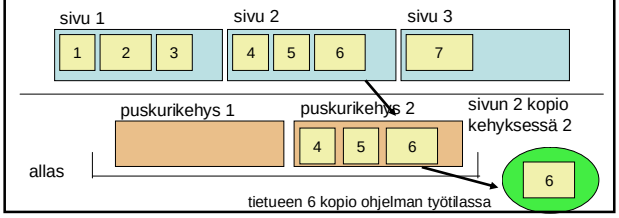
8

Tiedostojen käsittely

Tiedostorakennetason näkemys: tiedostossa on osoitettavissa olevia tietueita



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



Tiedostojen käsittely

Tietueen haku ohjelman käyttöön, karkea algoritmi:

```
tiedostorakenne.hae_tietue(tietuetunnus rid, muuttuja m)  {
   sivuosoite os =
        tiedostorakenne.päättele_sivu_tietueelle(rid);
   puskurikehys p = puskurinhallinta.anna_sivu(os);
   paikka offs=
      tiedostorakenne.päättele_sijainti(tietuelle rid,
            kehyksessä p);
      tiedostorakenne.siirrä(muutujaan m, kehyksestä p,
            kohdasta offs);
}
```

10

Tiedostojen käsittely

```
puskurikehys puskurinhallinta.anna_sivu(os) {
   puskurikehys q =
        puskurinhallinta.missä_kehyksessä_on_sivu(os);
   jos (q==null)  {     // ei ole missään
        q= puskurinhallinta.valitse_vapaa_puskuri();
        puskurinhallinta.lataa_sivu(puskuriin q sivu os);
        puskurinhallinta.naulitse(sivu os puskuriin q); }
   muuten  {
        puskurinhallinta.naulitse(sivu os puskuriin q);
   }
   palauta_osoitin_puskuriin q;
}
```

11

Tiedostojen käsittely

- Puskurinhallinnan on tiedettävä minkä sivun kopioita puskurikehyksissä on ja mitkä puskurikehykset ovat vapaina tiedonsiirtoa varten.
- Naulitsemismenettelyllä (fix, pinning) pidetään kirjaa varatuista puskurikehyksistä – naulitseminen kasvattaa kehyksen käyttäjälaskuria (pin counter). Kun sivun tarvitsija on käsitellyt sivun se ilmoittaa vapauttavansa puskurikehyksen (unfix), jolloin laskurin arvoa vähennetään. Kun yksikään prosessi ei enää tarvitse kehyksessä olevaa sivua, on kehys vapaa uudelleenkäyttöön.

12

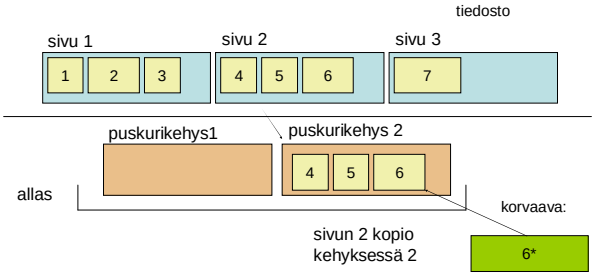
Tiedostojen käsittely

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

13

Tiedostojen käsittely

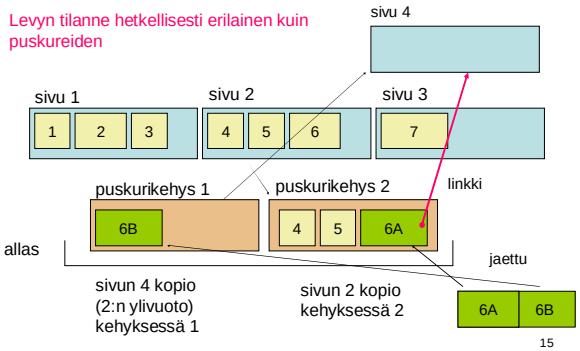
Tietueen pituus kasvaa:



14

Tiedostojen käsittely

Levyn tilanne hetkellisesti erilainen kuin puskureiden



15

Tiedostojen käsittely

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

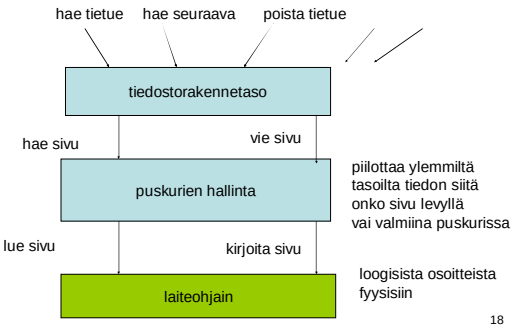
16

Tiedostojen käsittely

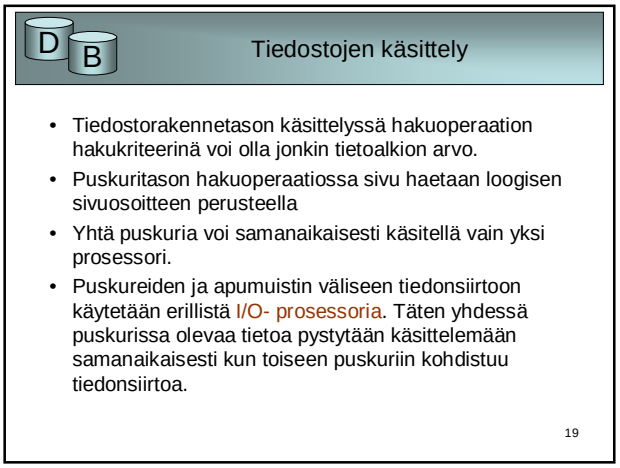
- Sivulla voi
  - olla useita tietueita tai
  - olla vain osa tietueesta (tietue voi jakautua useille sivuille).
- Sivu on abstraktio fyysisen levyn **lohko** (block) käsitteestä. Joissakin järjestelmissä on mahdollista siirtää kerralla useampien lohkojen **ryppäitä** (cluster). Tällöin sivu vastaisi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

17

Tiedostojen käsittely



18


D
B
Tiedostojen käsittely
• Tiedostorakennetason käsittelyssä hakuoperaation 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.
19