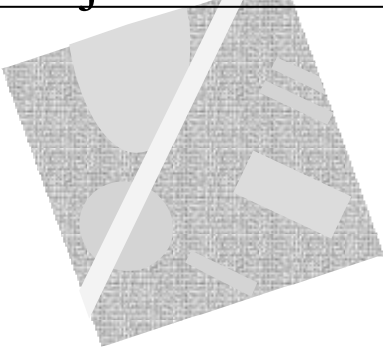


Jakso 8

Ohjelman toteutus järjestelmässä



Prosessi

PCB

I/O:n toteutus

29.5.2001Teemu Kerola, K20001

Prosessi ⁽⁴⁾

- Järjestelmässä olevan ohjelman esitysmuoto
- Järjestelmässä voi olla ”samalla kertaa” monta prosessia joko samasta tai eri ohjelmasta
 - käyttäjän (ihmisen) näkökulma ja aikaskaala (1 min, 1 sek?)
- suorittimella on yksi prosessi kerrallaan suorituksessa
 - laitteiston näkökulma ja aikaskaala (1 ms, 1 μs, 1 ns?)
- Muut prosessit ovat odottamassa jotakin
 - suorinta, I/O:ta, viestiä toiselta prosessilta

29.5.2001Teemu Kerola, K20002

Prosessin vaihto ⁽⁴⁾

- Suorittimella suoritusvuorossa olevan prosessin vaihtaminen
- Tapahtuu aika usein
 - keskimäärin noin 2000-3000 konekäskyn välein?
 - Esim. 50-500 kertaa sekunnissa?
- Iso operaatio - paljon kopiointia
 - satoja konekäskyjä

29.5.2001
Teemu Kerola, K2000
3

Prosessin elinkaari ⁽⁹⁾

```

    graph TD
        luonti((luonti)) --> valmis((valmis suoritukseen))
        valmis -- ready-to-run --> suorituksessa((suorituksessa))
        suorituksessa -- running --> valmis
        suorituksessa --> odottaa((odottaa))
        odottaa -- waiting --> valmis
        odottaa --> poistettu((poistettu tai tapettu))
    
```

Prosessin tilat järjestelmässä

29.5.2001
Teemu Kerola, K2000
4

Prosessin esitysmuoto järjestelmässä ⁽⁴⁾

- Prosessin kuvaaja tai kontrollilohko (PCB - process control block)
 - isohko tietue, joka sisältää kaiken yhdestä prosessista
 - muistialueet, tiedostot, tiedostojen käsittelykohdat
 - ei suorituksessa oleville myös: suorittimen tila (laiterekisterit, MMU:n rekisterit, kontrollirekisterit)
 - joka prosessista oma PCB
 - käyttöjärjestelmärutiinit manipuloivat PCB:tä

29.5.2001

Teemu Kerola, K2000

5

Prosessin tilanvaihdon toteutus ⁽⁹⁾

- Prosessin tilanvaihto tapahtuu siirtämällä prosessi (sen PCB) jonosta toiseen
 - ready-to-run jono (tai jonot)
 - running jono
 - jota ei oikeastaan ole olemassa
 - waiting jono
 - joka tyypille oma jononsa
 - esim: laitteen Disk I/O:n valmistumista odottavat
 - esim: näppäimistön painallusta odottavat
 - esim: kellolaitekeskeytystä odottavat
 - esim: prosessilta 1345 signaalia odottavat

29.5.2001

Teemu Kerola, K2000

6

Prosessit jonoissa (1)

R-to-R → 1056 → 1766 ⇨ Running → 0188

Disk1 → 0036 → 7654 → 9878

Timer → 0555

Msg from 1345 → 2222

Prosessin 9878 kuvaaja (PCB)

Vuoronanto:
valitse seuraava prosessi Ready-to-Run -jonosta ja siirrä se suoritukseen CPU:lle
 (kopioi tämän prosessin suorittimen tila suorittimelle)

29.5.2001 Teemu Kerola, K2000 7

KJ esimerkki: I/O keskeytys (5)

R-to-R → R → Q → Tdriver → Running → P

Disk1 → Tdriver

Timer → S

Msg from Tdriver → U

I/O keskeytys laitteelta Disk1 prosessille Tdriver?
 Suoritin havaitsee keskeytyssignaalin ja suorittaa I/O keskeytyssäätelyrutiinin (P:n ympäristössä)
 Tdriver siirretään R-to-R jonoon
 P:n suoritus jatkuu vai jatkuuko?

29.5.2001 Teemu Kerola, K2000 8

KJ esimerkki: I/O keskeytys (ei anim)

I/O keskeytys laitteelta Disk1 prosessille Tdriver?
 Suoritin havaitsee keskeytyssignaalin ja suorittaa I/O keskeytyksenkäsittelyrutiinin (P:n ympäristössä)
 Tdriver siirretään R-to-R jonoon
 P:n suoritus jatkuu vai jatkuuko?

29.5.2001 Teemu Kerola, K2000 9

KJ esim: aikaviipalekeskeytys (7)

P saa aikaviipalekeskeytyksen?
 P siirretään takaisin R-to-R jonoon
 Seuraava prosessi R saa suoritusvuoron
 Entä jos P olisi pyytänyt levy I/O:ta Disk1:ltä?

29.5.2001 Teemu Kerola, K2000 10

KJ esim: aikaviipalekesk. (ei anim)

R-to-R → R → Q → P

Running → P

waiting {

Disk1 → Tdriver

Timer → S

Msg from Tdriver → U

P saa aikaviipalekeskeytyksen?
 P siirretään takaisin R-to-R jonoon
 Seuraava prosessi R saa suoritusvuoron
 Entä jos P olisi pyytännyt levy I/O:ta Disk1:ltä?

29.5.2001
Teemu Kerola, K2000
11

Prosessin kuvaajan sisältö (9)

- Prosessin tunniste 14023
- Prioriteetti suorittimen vuoronantoa varten 143
- Prosessin tila ja odottamisen syy R-to-R
- Suoritinympäristö talletettuna odottamisen aikana
 - rekisterit, PC, SP, FP, tilarekisterit
- Prosessin ensimmäisen käskyn osoite Main { }
- Poikkeuskäsittelijöiden osoitteet
- Aikaviipale
- Käytössä olevat muistialueet, aukiolevat tiedostot
- KJ:n hallintotietoa (kokonaisaika, etc etc)

29.5.2001
Teemu Kerola, K2000
12

Prosessin vaihto ⁽⁴⁾

- Vaihdon tekee KJ rutiini sillä hetkellä suorittavan prosessin ympäristössä
- Talleta vanhan prosessin suoritin ympäristö suorittimelta omalle talletusalueelle muistiin
 - talleta kaikki suorittimella olevat tiedot muistiin
- Kopio uuden prosessin suoritin ympäristö omalta talletusalueeltaan suorittimelle
 - lataa kaikki suorittimen rekisterit (myös PC!)
- Uuden prosessin suoritus jatkuu täsmälleen siitä mihin viime kerralla jäätin
 - sama konekäsky, käytännössä sama suoritusympäristö!
 - Usein keskellä prosessin vaihtoa suorittavaa KJ-

29.5.2001 rutiiniinia

Teemu Kerola, K2000

13

Prosessin prioriteetti ⁽³⁾

- Prosessin tärkeysjärjestys suorittimella
 - esim. pieni numero \Rightarrow iso prioriteetti
- Joka prioriteetti (luokalle) oma R-to-R jononsa
 - KJ prosesseilla parempi prioriteetti kuin käyttäjätason prosesseilla
 - tosiaikasovelluksen prosesseilla parempi prioriteetti kuin KJ prosesseilla
- Prioriteetti voi vaihdella prosessin elinaikana
 - paljon suoritinaikaa \Rightarrow huonompi prioriteetti
 - kauan R-to-R jonossa \Rightarrow parempi prioriteetti (prosessi siirretään korkeamman prioriteetin R-to-R jonoon)

29.5.2001

Teemu Kerola, K2000

14

Käyttöjärjestelmän toteutus ⁽⁵⁾

- Joukko prosesseja ja proseduureja
 - prosessit elävät omaa elämäänsä (etuoikeutetussa tilassa eli root'ina?) koko ajan
 - swapper (Unix) - muistinhallintaprosessi
 - init prosessi (Unix) - kaikkien käyttäjätason prosessien "äiti"
 - laiteajurit
 - proseduurit suoritetaan sen hetkisen prosessin ympäristössä (etuoikeutetussa tilassa?)
 - keskeytyskäsitelijät
 - Saavat kontrollin aina tarvittaessa
 - keskeytykset, SVC, ajastimet

29.5.2001

Teemu Kerola, K2000

15

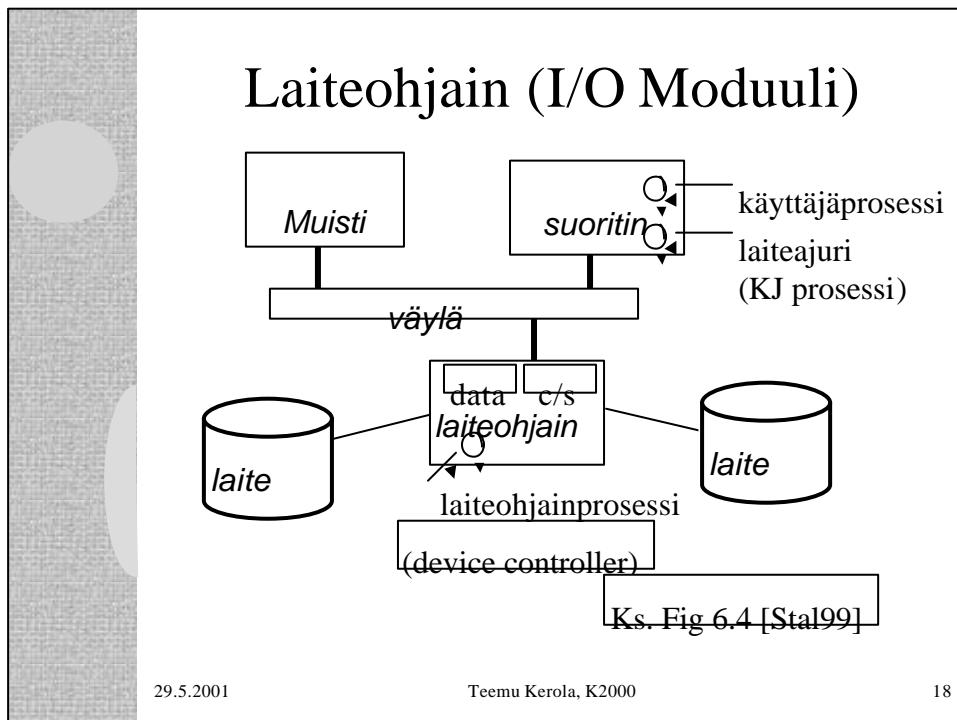
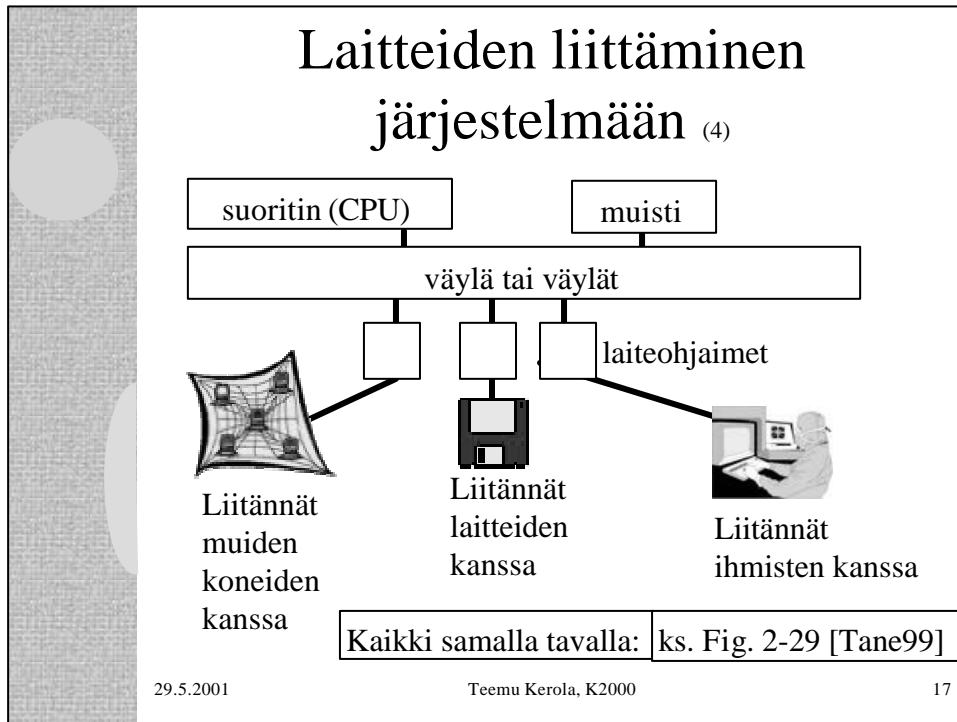
KJ esimerkki: laiteajuri

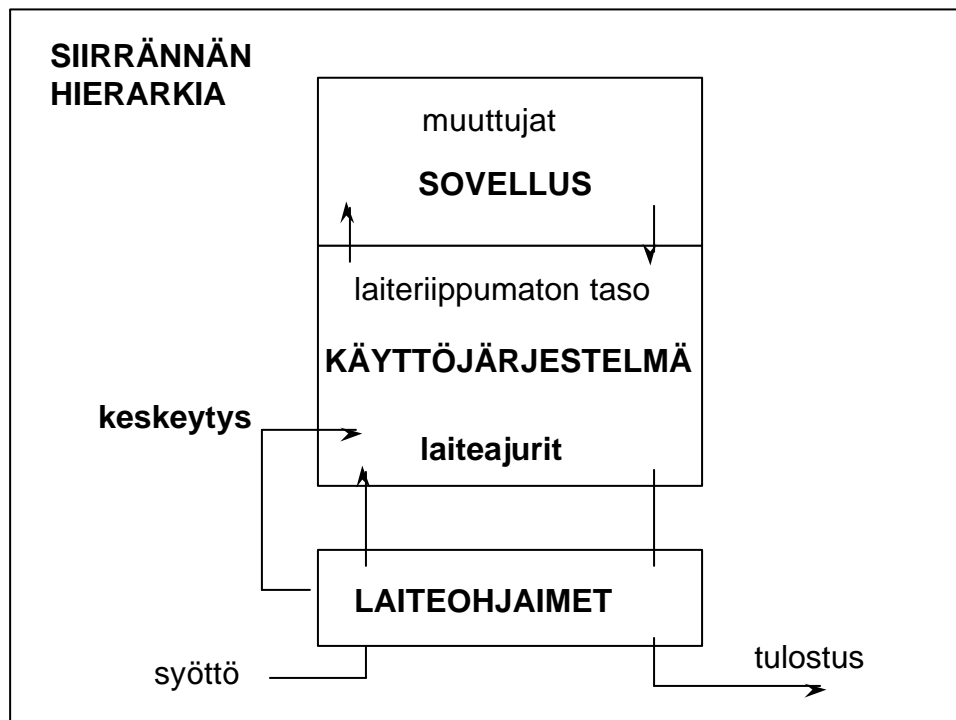
- Prosessina
 - proseduurina kutsuttu laiteajurin tynkä (stub) lähettää I/O-pyynnön viestinä laiteajurille ja odottaa vastausta
 - tynkä voi olla käyttäjätasoinen
 - ajuriprosessi voi olla (joskus) etuoikeutettu
 - vaatii prosessien välistä viestintää
 - Proseduurina
 - laiteajuri suoritetaan KJ-rutiinina tavallisen aliohjelmakutsun tai SVC-kutsun kautta
 - osa tai kaikki koodista voi olla etuoikeutettua
 - vain yksi kutsu kerrallaan suorituksessa?

29.5.2001

Teemu Kerola, K2000

16





- Siirränän hierarkia

- **sovellusohjelmataso**

- loogisia kokonaisuuksia, tietueita ja tiedostoja

- ohjelman sisäisiä nimiä

- Readln (File1, X)

- Open (Tdsto, RW)

- käyttöjärjestelmätaso

- laitteisto

- Siirrännän hierarkia
 - sovellusohjelmataso
 - **käyttäjärjestelmätaso**
 - rutiinit, jotka toteuttavat ja valvovat siirräntää
 - laiteriippumaton siirräntä
 - sovellukselle yhtenäinen tapa käyttää kaikkia siirräntäpalveluita
 - laiteriippuva siirräntä
 - laitteiden todelliseen käyttöön liittyvä ohjausohjelmisto
 - koodattu **laiteajureihin**
 - laitteisto

- Siirrännän hierarkia
 - sovellusohjelmataso
 - käyttäjärjestelmätaso
 - **laitteisto**
 - siirräntä voidaan toteuttaa kokonaan prosessorin valvonnassa
 - ei hyödynnetä rinnakkaisuutta
 - **laiteohjain** (siirräntään erikoistunut prosessori) huolehtii itsenäisesti siirrännästä
 - prosessorin ja ohjainten välinen kommunikointi

- laitekuvaaja

- yksi kutakin laitetyyppiä varten
- talletettavat tiedot riippuvat laitteesta
 - laitteen yksilöivä tunnus (väyläosoite)
 - ohjeet laitteen käytöstä
 - urien, sektorien ja levypintojen määrä, lohkon koko
 - viitteet näppäimistön merkinmuunnostauluihin
 - laitteen tilatietoa
 - varattu/vapaa/rikki
 - viitteet jonottaviin palvelupyyntöihin
 - viite laitetta käyttävän prosessin kuvaajaan

- laiteriippumattoman siirrän tehtäviä

- loogisesta tiedostonimestä => käytettävän laitteen tyyppi
- pitää kirjaa levytilan vapaista ja varatuista alueista
- siirrän puskurointi (levylohko)
- luku/kirjoituskohdan ylläpito
- tarvittaessa käynnistää fyysisen siirrän
 - antaa laiteajurille tehtävän

- laiteajurin tehtäviä

- tehtävät riippuvat laitteesta

- muodostaa parametrien ja laitekuvaajan perusteella laitetta ohjaavat käskyt
 - esim. levylohkonumeroiden muuttaminen levypinnan, uran ja sektorin numeroiksi
- levypyyntöjen optimointi
- ohjaimella tehtävän fyysisen siirännän käynnistys
- siirännän kirjanpito
- siirron oikeellisuuden tarkistukset ja virheiden korjausyritykset

Laitteiden käytön toteutus ⁽⁵⁾

ks. laiteohjainkuva

- Käyttäjäohjelma kutsuu käyttöjärjestelmän laiteajuria tekemään I/O:n. Laiteajuri suoritetaan samalla suorittimella kuin käyttöohjelmakin.
- Laiteajuri ohjaa laitteen toimintaa laitteen laiteohjaimella olevien kontrollirekisterien (muistialue) avulla
- Laiteajuri voi lukea laitteen tilatietoa laiteohjaimella olevien statusrekisterien (muistialue) avulla
- Laiteajuri voi lukea (kirjoittaa) laitteen lukemaa (laitteelle kirjoitettavaa) tietoa laiteohjaimella olevien datarekistereiden (muistialue) avulla
- Kontrolli-, status- ja datarekisteri kolmikko muodostaa ”I/O portin” suorittimen näkökulmasta

Laiteohjaimen rekistereihin viittaaminen ⁽⁵⁾

- Ongelma: miten suorittimella ks. laiteohjainkuva suorittavan laiteajuri viittaa eri kortilla oleviin rekistereihin?
- Ratkaisu 1: omat I/O-konekäskyt tätä tarkoitusta varten
 - käskyssä annetaan laiteohjaimen identifikaatio ja rekisterin nro (I/O osoiteavaruus)
 - vaikea laajentaa käyttöä uusiin laitteisiin, joilla rekisterit voivat olla hyvinkin erilaisia

| | |
|------|----------------------|
| x86: | IN, OUT INS, OUTS |
|------|----------------------|

| | |
|--------|------------------------------|
| KOKSI: | IN R1, =KBD, OUT R2, =CRT |
|--------|------------------------------|

29.5.2001

Teemu Kerola, K2000

27

Ratkaisu 2: muistiinkuvattu I/O ⁽⁵⁾

- Laiteajuri lukee/kirjoittaa laiteohjaimella olevia rekistereitä (data, status/kontrolli) tavallisilla muistin luku/kirjoitus käskyillä
 - ei tarvita erillisiä I/O-konekäskyjä!
 - laiteohjaimella olevat ”rekisterit” ovat samanlaista viitattavaa muistia kuin ”normaali muisti”
 - muistisoitteen ensimmäiset bitit valitsevat, mille laitteelle (vai tavallisen muistiin) viittaus kohdistuu
 - voidaan käyttää rinnan I/O käskyjen kanssa (laiterekistereihin voi siis viitata sekä I/O-käskyillä että muistiinkuvatun I/O:n avulla)

| |
|------------------------------|
| esim. Intelin arkkitehtuurit |
|------------------------------|

29.5.2001

Teemu Kerola, K2000

28

I/O tyypit ⁽²⁾

ks. laiteohjainkuva

- Suora I/O: laiteajuri odottaa tiukassa silmukassa, kunnes laiteohjaimen statusrekisteri ilmoittaa I/O-pyyntöön valmistuneen (direct I/O)
 - laiteajuri siirtää tietoa muistin ja datarekisterin välillä
- Epäsuora I/O: I/O:n odotusaikana suorittimella suoritetaan jotain muuta ohjelmaa (indirect I/O interrupt driven I/O)
 - Kun I/O-pyyntö valmistuu, laiteohjain antaa keskeytyksen (laitekeskeytys, I/O interrupt) suorittimelle, joka (jonkin ajan kuluttua) jatkaa kesken jäänyttä I/O-pyyntöön esittänyttä ohjelmaa.
 - laiteajuri siirtää tietoa muistin ja datarekisterin välillä

29.5.2001

Teemu Kerola, K2000

29

I/O tyypit (jatkoa) ⁽⁴⁾

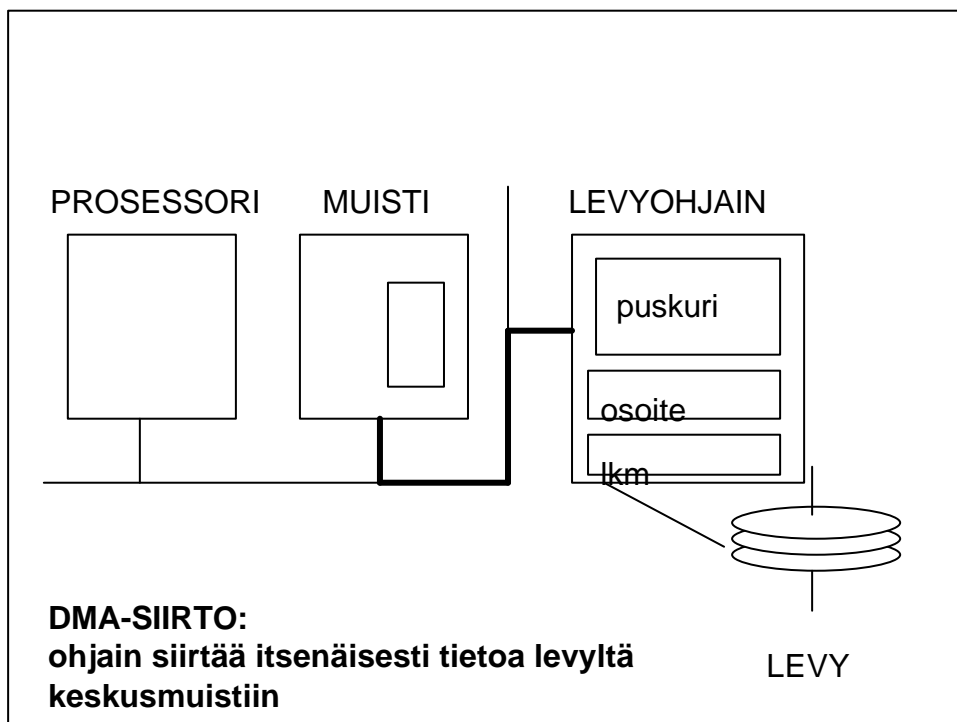
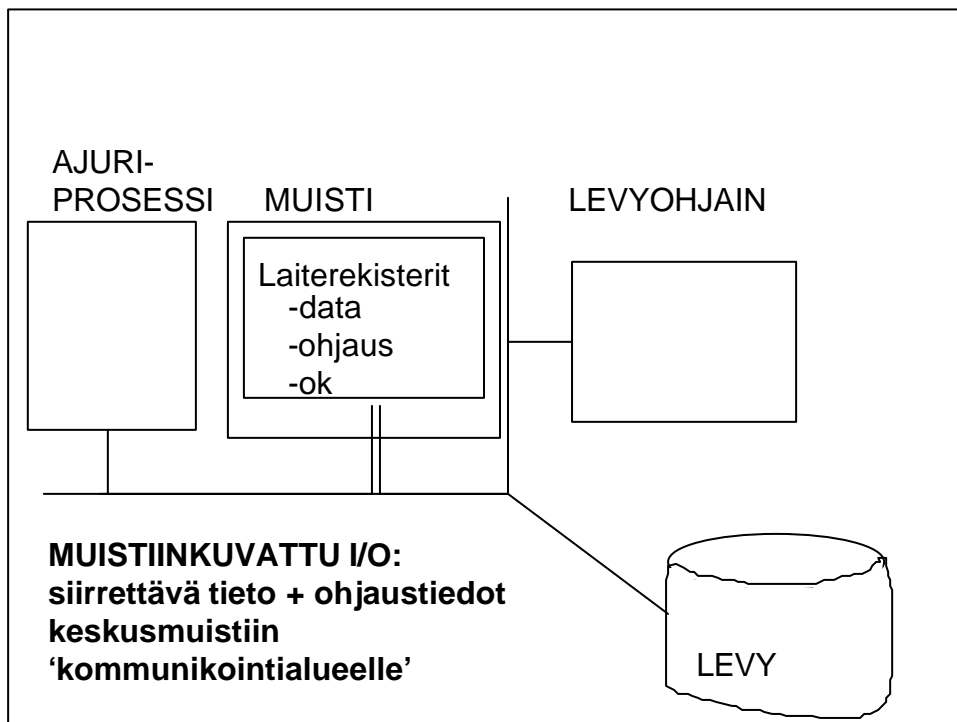
ks. laiteohjainkuva

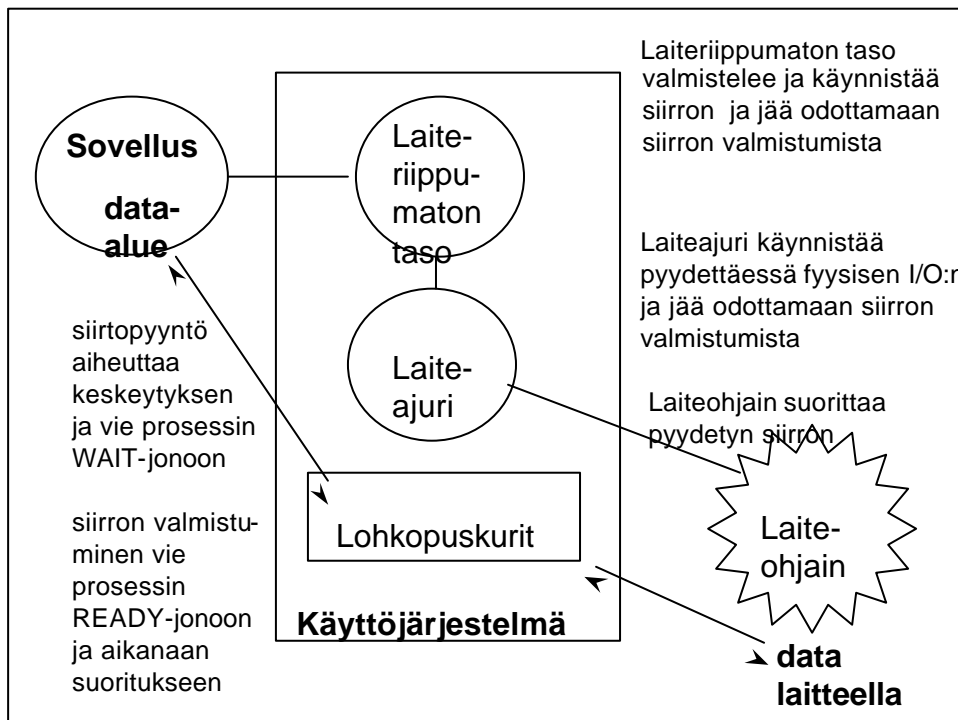
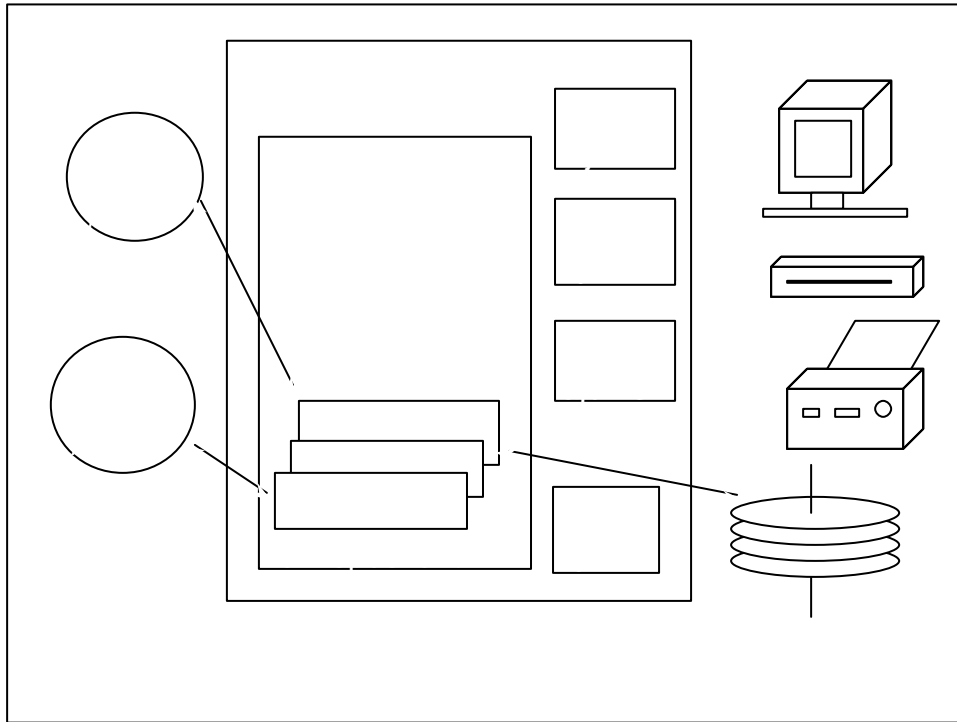
- DMA - Direct Memory Access
 - älykkäämpi laiteohjain
 - laiteohjain voi suoraan kopioida tiedot keskusmuistiin (laiteajurin ei tarvitse siirtää tietoa muistin ja datarekisterin välillä)
 - laiteohjain tekee paljon suuremman määrän työtä itsenäisesti (kuin epäsuorassa I/O:ssa) ennen suorittimelle annettavaa laitekeskeytystä

29.5.2001

Teemu Kerola, K2000

30





-- Jakson 8 loppu --

[Tane99]

```
// Open files for input and output.
inhandle = CreateFile("data", GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
outhandle = CreateFile("newf", GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);

// Copy the file.
do {
    s = ReadFile(inhandle, buffer, BUF_SIZE, &count, NULL);
    if (s > 0 && count > 0) WriteFile(outhandle, buffer, count, &ocnt, NULL);
    while (s > 0 && count > 0);

// Close the files.
CloseHandle(inhandle);
CloseHandle(outhandle);
```

Figure 6-40. A program fragment for copying a file using the Windows NT API functions. This fragment is in C because Java hides the low-level system calls and we are trying to expose them.

Lisää tietoa?  KJ kurssit, RIO, HajJärj