

Kertausluento 3 (lu7, lu8)  
Tiedon muuttumattomuuden tarkistus  
Järjestelmän sisäinen muisti  
Ohjelman toteutus järjestelmässä

Ohjelman esitysmuoto  
Rakenteellinen tieto  
Pariteetti, Hamming-koodi  
Välimuisti, muisti

Prosessi, sen esitysmuoto  
Käyttöjärjestelmä

19.4.2010

Copyright Teemu Kerola 2010

1

## Konekäskyjen esitysmuoto muistissa

- Konekohtainen, jokaisella omansa
- Käskyt ovat 1 tai useamman tavun mittaisia
  - SPARC, kaikki käskyt: 1 sana eli 4 tavua
  - PowerPC, kaikki käskyt: 1 sana eli 4 tavua
  - Pentium II: 1-16 tavua, paljon variaatioita
- Käskyillä on yksi tai useampi muoto, kussakin tietty määrä erilaisia kenttiä
  - opcode, Ri, Rj, Rk, osoitusmoodi
  - pitkä tai lyhyt vakio

TTK-91, kaikki käskyt: 1 sana, 1 muoto

19.4.2010

Copyright Teemu Kerola 2010

2

## Taulukkojen esitysmuoto

- Peräkkäisrakenteena, kuten esimerkit aikaisemmin
- Riveittäin tai sarakeittain
- Ei omia konekäskyjä, manipulointi aliohjelmilla tai loopeilla  
Poikkeus: vektorisuorittimet, joilla
  - vektorirekisterit (esim. 64 liukulukua) tavallisten rekistereiden lisäksi
  - omia konekäskyjä vektoriopeaatioita varten
- Indeksoitu tiedonositusmoodi tukee 1-ulotteisten taulukoiden käyttöä

19.4.2010

Copyright Teemu Kerola 2010

3

## Tietueiden esitysmuoto

- Peräkkäisrakenteena
- Osoite on jonkin osoitinmuuttujan arvo
- Ei omia konekäskyjä, manipulointi aliohjelmilla tai kääntäjän generoimien vakiolisäysten avulla
- Indeksoitu tiedonositusmoodi tukee tietueiden käyttöä

19.4.2010

Copyright Teemu Kerola 2010

4

## Olioiden esitysmuoto

- Kuten tietueet, yleensä varattu keosta (heap)
- Useat olion kentistä sisältävät vuorostaan osoitteen keosta suoritusaikana varattuun toiseen olioon
- Metodit ovat aliohjelmien osoitteita
- Ei omia konekäskyjä, manipulointi aliohjelmilla

19.4.2010

Copyright Teemu Kerola 2010

5

## Tiedon tarkistus

- Tiedon oikeellisuutta ei voi tarkistaa yleisessä tapauksessa
- Laitteistovirheitä voidaan havaita ja joskus automaattisesti korjata
  - bitti voi muuttua muistissa tai tiedon siirrossa
    - muistipiirissä voi olla vika (staattinen vika)
    - sopiva alkeishiukkanen voi muuttaa bitin tiedonsiirron aikana (transientti virhe)
  - korjaamattomasta virheestä voi aiheutua häiriö
- Tietokannan eheys on eri asia!

Lisää  
tietoa?Tieto-  
kanta  
kurssit

19.4.2010

Copyright Teemu Kerola 2010

6

## Tiedon muuttumattomuus

- Perusidea: otetaan mukaan ylimääräisiä bittejä, joiden avulla virheitä voidaan havaita ja ehkä myös korjata
- Järjestelmä suorittaa tarkistukset automaattisesti joko laitteistotasolla tai ohjelmiston avulla

19.4.2010

Copyright Teemu Kerola 2010

7

## Bittitason tarkistukset

- Muistipiirit, levyt, väylät, tiedonsiirrot 120464-121C  
(merkkejä, ei bittejä)
- Monenko bitin muuttuminen Hetu: 1  
havaitaan?
- Monenko bitin muuttuminen Hetu: 0  
voidaan automaattisesti korjata?
- Havaitsemiseen ja/tai korjaamiseen tarvitaan enemmän (ylimääräisiä) bittejä
  - lisämuistitilan tai levytilan tarve? Hetu: +10%
  - lisäpiuhojen tarve väylällä?
- Tarkistukset/korjaukset Hetu: ohjelmistotasolla  
laitteisto- vai SW-tasolla?

19.4.2010

Copyright Teemu Kerola 2010

8

## Pariteettibitti

- Yksi ylimääräinen bitti per tietoalkio
  - sana, tavu, tietoliikennepaketti
- Parillinen (pariton) pariteetti: 1-bittien lukumäärä on aina parillinen (pariton)
- Havaitsee: 1 bitti
- Korjaa: 0 bittiä
- Esimerkki (parillinen pariteetti)

0010 001 0

1000 1101 1111 001 1

19.4.2010

Copyright Teemu Kerola 2010

9

## Hamming etäisyys

- Montako bittiä jossain koodijärjestelmässä (esim ISO Latin-1) esitetyllä koodilla (esim. 'A' = 0x41 = 0100 0001) täytyy muuttua, että se muuttuu johonkin toiseen (mihin tahansa) lailliseen koodiin.

'A' = 0x41 = 0100 0001

2 bittiä

'B' = 0x42 = 0100 0010

1 bittiä

'C' = 0x43 = 0100 0011

- ISO Latin-1:n Hamming etäisyys: 1
- Pariteettibitin kanssa Hamming etäisyys: 2
  - mikä todennäköisyys 2 bitin (vs. 1 bitin) virheeseen?
  - riittävän pieni?  $(\text{Prob}\{ \text{"yhden bitin virhe"} \})^2$

19.4.2010

Copyright Teemu Kerola 2010

10

## Virheen korjaava Hamming koodi<sup>(6)</sup>

**Figure 2-14.** (a) Encoding of 1100. (b) Even parity added. (c) Error in AC.

[Tane99]

(a) Kukin databitti (4 kpl) kuuluu erilaisiin pariteettijoukkoihin (3 kpl)

(b) Tarvitaan 3 ”ylimääräistä” bittiä!

(c) Joukot A ja C havaitsevat virheen ja siten paikallistavat virheellisen bitin

Täsmälleen 1 databitti identifioituu kerrallaan!

entä jos virhe pariteettibitissä?

19.4.2010 Copyright Teemu Kerola 2010 11

## Virheen korjaava Hamming koodi

Data:      100 1100      110 1100      (parillinen pariteetti)

Bitti nro:    765 4321      765 4321

	1
Pariteettibitti 1 tarkistaa bittejä 1, 3, 5, 7	1
Pariteettibitti 2 tarkistaa bittejä 2, 3, 6, 7	11
Pariteettibitti 4 tarkistaa bittejä 4, 5, 6, 7	1
	1 1
	11
	111

Tapahtuu virhe: bitti 6 muuttuu (flips)

Pariteettibitti 2 tarkistaa bittejä 2, 3, 6, 7: VIRHE

Pariteettibitti 4 tarkistaa bittejä 4, 5, 6, 7: VIRHE

2+4 = 6 ⇒ korjaa bitti nro 6

19.4.2010 Copyright Teemu Kerola 2010 12

## CRC - Cyclic Redundancy Code

- Tiedonsiirrossa käytetty tarkistusmenetelmä
- Tarkistussumma (16 bittiä) isolle tietojoukolle
  - laske  $CRC = f(\text{viesti}) \% 2^{16}$  (ota 16 viimeistä bittiä)
  - lähetä viesti ja CRC
  - vastaanota viesti ja CRC
  - laske CRC ja tarkista, oliko se sama kuin viestissä
  - jos pielessä, niin pyydä uudelleenlähetystä

### CRC-CCITT CRCs detect:

All single- and double-bit errors  
All errors of an odd number of bits  
All error bursts of 16 bits or less  
In summary, 99.998% of all errors

19.4.2010

Copyright Teemu Kerola 2010

13

## Virheiden tarkistusmenetelmien käyttöalueet

- Mitä lähempänä suoritinta, sitä tärkeämpää tiedon oikeellisuus on
- Sisäinen väylä, muistiväylä
  - virheet lennossa korjaava Hamming koodi
- Paikallisverkko
  - uudelleenlähetyksen vaativa CRC
  - kun tulee virheitä, niin niitä tulee yleensä paljon
    - Hamming koodi ei riitä kuitenkaan
    - pariteettibitti päästää läpi (esim.) 2 virheen paketit

19.4.2010

Copyright Teemu Kerola 2010

14

## Laitteiden monistaminen (kesk.)

- Monta muistipiiriä tai levyä, samat tiedot monistettu
- Monta suoritinta, samat käskyjen suoritukset monistettu
- Monta laitteistoa, samat ohjelmat monistettu
  - äänestysmenettely: enemmistö voittaa
  - monimutkainen, hidas?
  - virheelliseksi havaittu laitteisto suljetaan pois häiriköimästä automaattisesti?
- Eri tai saman tyyppiset laitteistot, samankaltaiset ohjelmat
  - samat speksit, samat syötteet, eri ohjelmoijat

“Four of the five computers (IBM AP-101) on the Columbia ran identical software and compared results with each other before giving the go-ahead to take a specific action. The fifth computer (also IBM AP-101) ran a different version of the software and was used only if the others failed.”

<http://www.hq.nasa.gov/office/pao/History/computers/contents.html>

19.4.2010

Copyright Teemu Kerola 2010

15

## Välimuisti (cache)

- Ongelma: keskusmuisti on aika kaukana suorittimesta
  - rekisterin viittausaika: X
  - muistin viittausaika: 10X
- Ratkaisu: välimuisti lähelle suoritinta
  - pidetään siellä (kopioita) viime aikoina viitatuista keskusmuistin alueista
    - välimuistin viittausaika: 2X
- Jokainen muistiviite on nyt seuraavanlainen
  - jos data ei ole välimuistissa, niin hae se sinne
    - suoritin odottaa tällä aikaa, laitteistototeutus!
  - tee viittaus dataan (käskyyn) välimuistissa
  - (talleta muutettu tieto keskusmuistiin)

19.4.2010

Copyright Teemu Kerola 2010

16



## Muistin toteutus

- Eri teknologioita eri tasoisiin muisteihin
- RAM - Random-Access Semiconductor Memory
  - anna osoite ja lue/kirjoita signaali
  - mistä vaan voi lukea/kirjoittaa samassa ajassa
  - virta pois  $\Rightarrow$  tiedot häviävät (volatile memory)

Huom: kaikki nykyiset muistit ovat ”random access”

19.4.2010

Copyright Teemu Kerola 2010

17

## RAM:n kaksi eri teknologiaa

- DRAM: dynaaminen RAM, halvempi, hitaampi, tietoja pitää virkistää vähän väliä (esim. joka 2 ms)
  - tavallinen keskusmuisti (1975-..) useimmissa koneissa
  - toteutettu kondensaattoreilla, jotka ”vuotavat” ...
- SRAM: staattinen RAM, kalliimpi (~10-20x), nopeampi (~10x), ei vaadi tietojen virkistämistä
  - välimuisti useimmissa koneissa
  - muisti superkoneissa (esim. Cray C-90)
  - toteutettu samanlaisilla logiikkaportteilla (gate) kuin prosessorikin

19.4.2010

Copyright Teemu Kerola 2010

18

## ROM teknologia

- ROM - Read-Only Memory
  - tieto säilyy virran katkettua (non-volatile)
  - voi käytössä vain lukea, ei voi kirjoittaa
    - esim. järjestelmän alustustiedot (BIOS)
  - kirjoitus lastun valmistusaikana, Mask-ROM
    - ei enää käytössä
  - huono puoli: kerran väärin, aina väärin (ehkä...)
  - päivitys: laita valmistajalta saatu uusi lastu paikalleen
  - tietoa voi lukea mistä vain samassa ajassa (random access)
  - yleensä hitaampi kuin RAM (~10x)

19.4.2010

Copyright Teemu Kerola 2010

19

## Kirjoitettavia ROM-muisteja

- PROM - Programmable ROM
  - kerran kirjoitettava
  - tiedon päivitys: ”polta” tiedot tyhjään PROM:iin
- EPROM - Erasable PROM
  - tietoja ei voi päivittää sana kerrallaan
  - vanhat tiedot voidaan (kaikki!) poistaa 20 min. UV-säteilyllä, jonka jälkeen päivitettyt tiedot voidaan ladata
- EEPROM - Electronically Erasable PROM
  - tietojen pyyhkiminen tavukohtaisesti elektronisesti
- FLASH EEPROM memory
  - tietojen pyyhkiminen nopeasti kerralla elektronisesti
  - normaalijännitteellä, kaikki tai lohko kerrallaan
  - nopeampi kuin EEPROM

read-mostly memory

19.4.2010

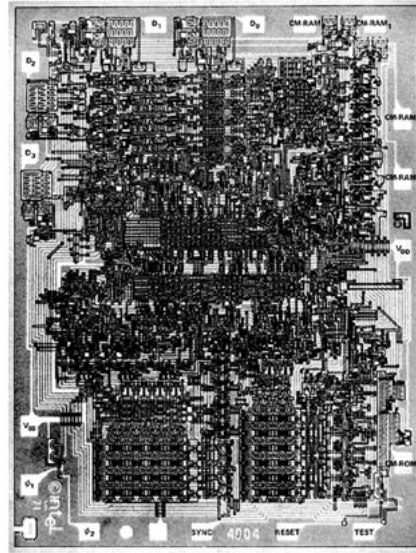
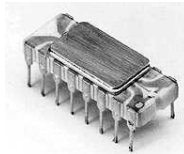
Copyright Teemu Kerola 2010

20

## -- Luennon 7 loppu --

### Intel 4004, 1971

- Faggin, Hoff, Mazor
- Ens. suoritin lastulla  
3x4 mm, \$200
- 2300 transistoria
- 4 bitin sana
- Laskinta varten
- Sama laskentateho kuin  
Eniacilla  
(18000 tyhjiöputkea)



19.4.2010

Copyright Teemu Kerola 2010

21

19.4.2010

Copyright Teemu Kerola 2010

22

# Luento 8

## Ohjelman toteutus järjestelmässä

Prosessi  
Prosessin esitysmuoto järjestelmässä  
Käyttöjärjestelmä  
KJ-prosessit

19.4.2010

Copyright Teemu Kerola 2010

23

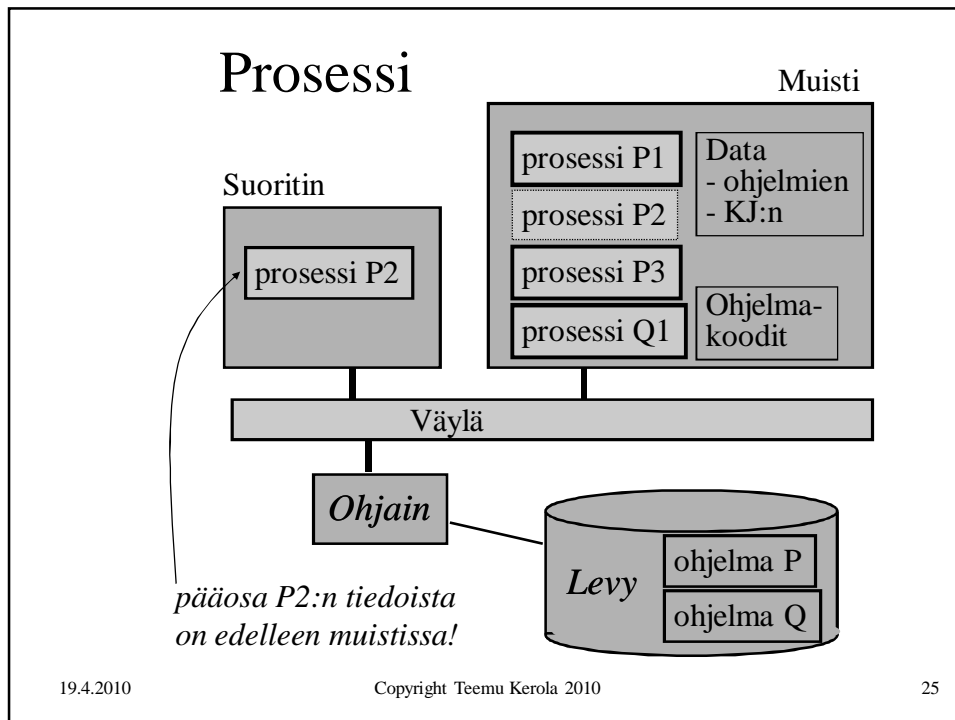
## Prosessi

- 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 suorituksessa on yksi prosessi kerrallaan
  - laitteiston näkökulma ja aikaskaala (1 ms, 1  $\mu$ s, 1 ns?)
- Muut prosessit ovat odottamassa jotakin
  - suorittinta? I/O:ta? viestiä toiselta prosessilta?
  - vapaata muistitilaa?

19.4.2010

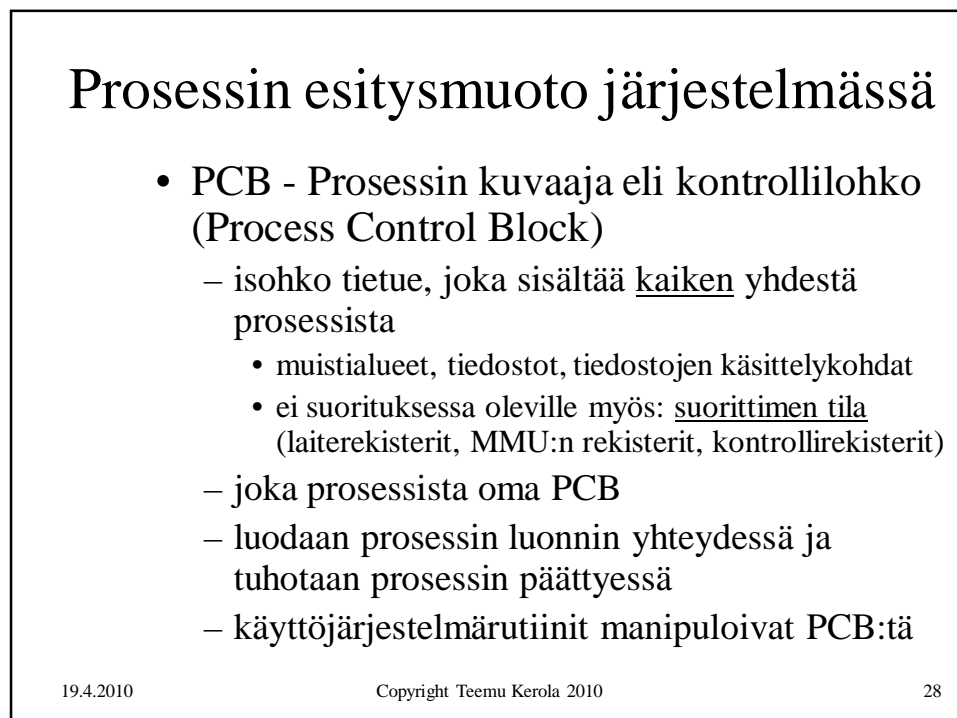
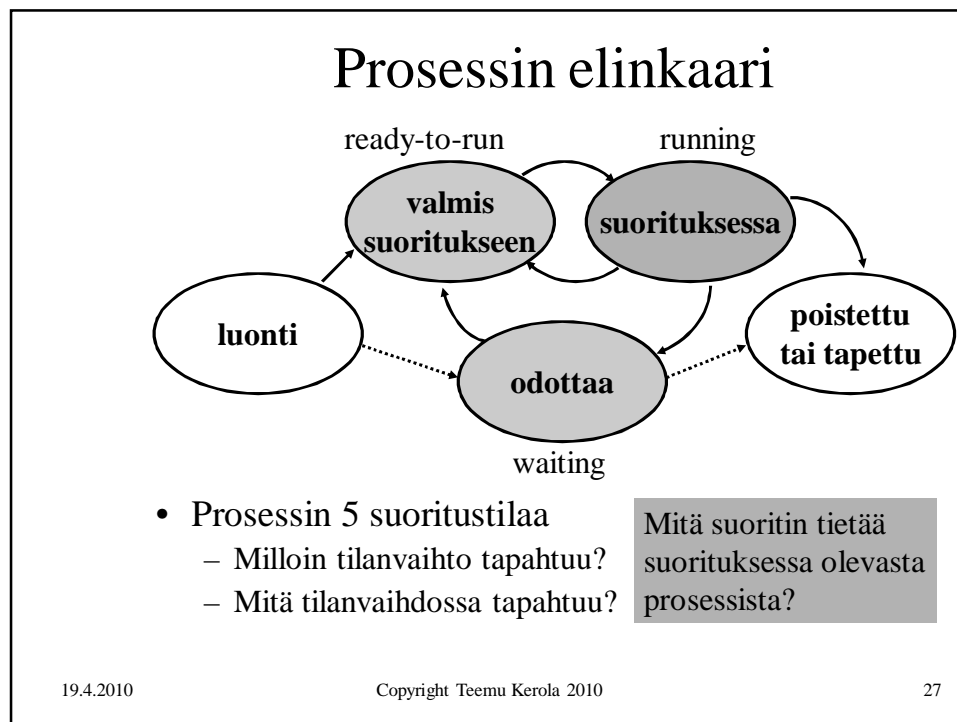
Copyright Teemu Kerola 2010

24



## Prosessin vaihto

- Suorittimella suoritusvuorossa olevan prosessin vaihtaminen
- Tapahtuu aika usein
  - keskimäärin noin 2000-3000 konekäskyn välein?
  - esim. 50-500 kertaa sekunnissa?
- Iso operaatio - paljon kopiointia
  - montako konekäskyä tähän kuluu? 50-500?  
0?



## Prosessin kuvaajan sisältö

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

19.4.2010

Copyright Teemu Kerola 2010

29

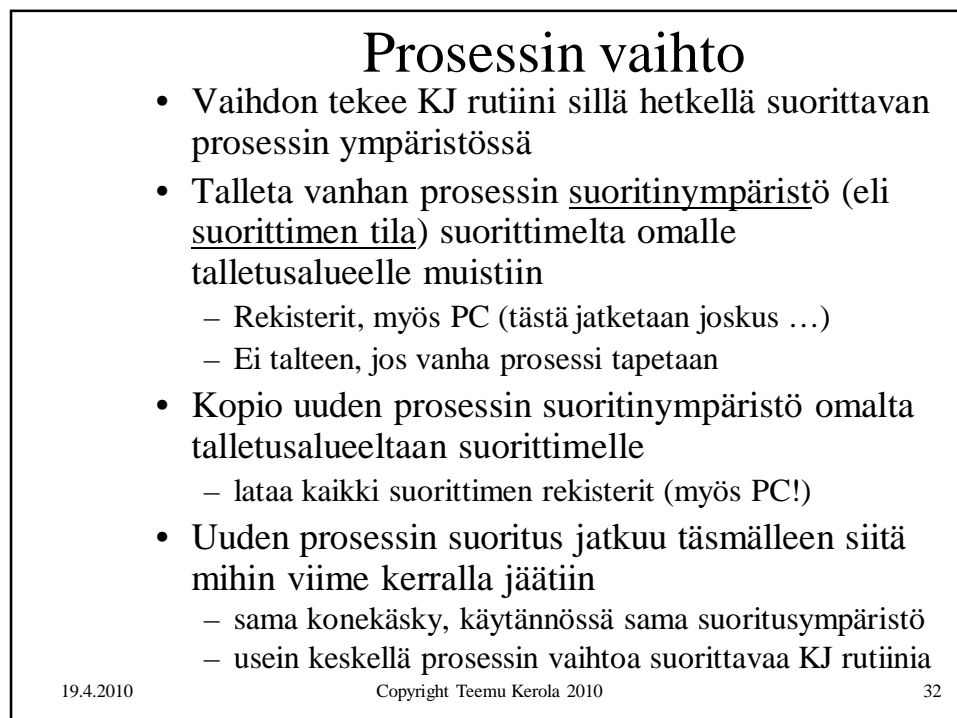
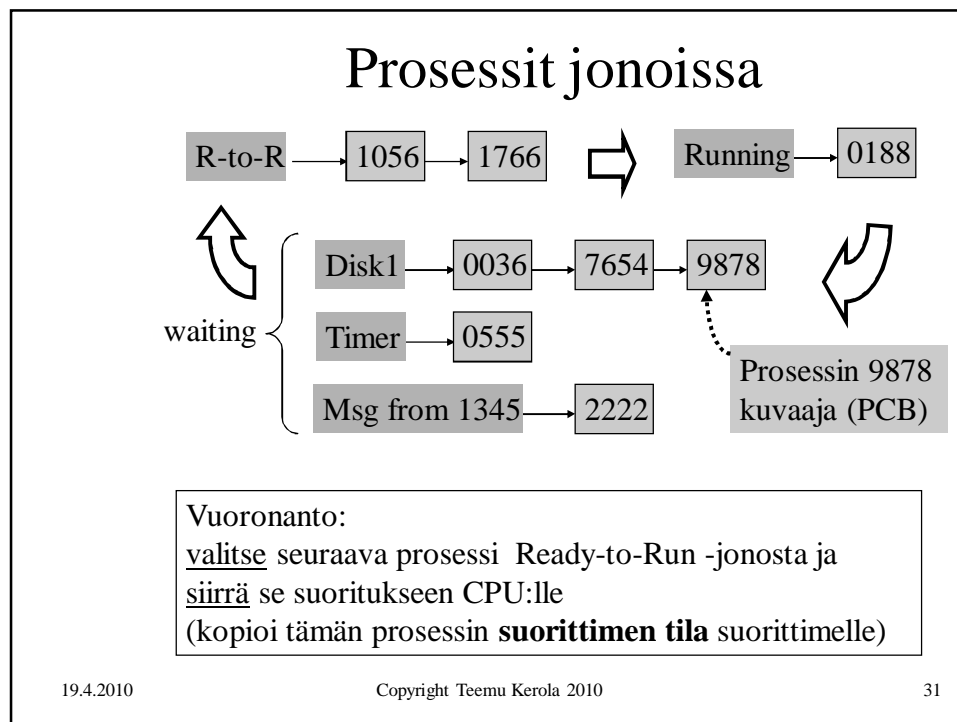
## Prosessin tilanvaihdon toteutus

- Prosessin tilanvaihto tapahtuu siirtämällä prosessi (sen PCB) jonosta toiseen
  - ready-to-run jono (tai jonot) odottaa suoritinta
  - running jono suorituksessa
    - ei oikeastaan ole olemassa
  - waiting jono odottaa jotakin
    - joka tyypille oma jononsa
    - esim: laitteen Disk1 I/O:n valmistumista odottavat
    - esim: näppäimistön painallusta odottavat
    - esim: kellolaitekeskeytystä odottavat
    - esim: prosessilta 1345 signaalia odottavat

19.4.2010

Copyright Teemu Kerola 2010

30





## Prosessin prioriteetti

(kesk.)

- **Prosessin tärkeysjärjestys suorittimella**
  - esim. pieni numero  $\Rightarrow$  iso (parempi) prioriteetti
- **Joka prioriteetti(luokalle) oma R-to-R jononsa**
  - KJ prosesseilla parempi prioriteetti kuin käyttäjätason prosesseilla
  - tosiaikasovelluksen prosesseilla parempi prioriteetti kuin KJ prosesseilla
    - muistakaa antaa KJ:lle aikaa aina joskus .... !
- **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

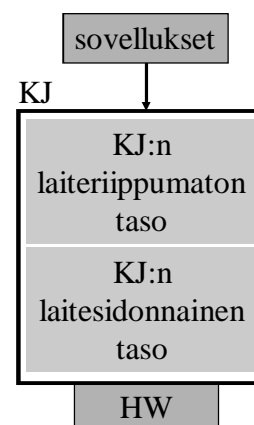
19.4.2010

Copyright Teemu Kerola 2010

33

## Käyttöjärjestelmän tavoitteet

- **Laiteriippumaton (HW-riippumaton) käyttöliittymä laitteistoon**
  - järjestelmää on helppo käyttää
  - järjestelmä antaa reilua palvelua kaikille
  - sovellukset on helppo tehdä
  - sovellukset on helppo siirtää muista järjestelmistä



19.4.2010

Copyright Teemu Kerola 2010

34

## Käyttöjärjestelmän tavoitteet (jatk)

- Järjestelmän resurssien tehokas hallinta
  - kaikista resursseista saada maksimihyöty
    - kuka osti liikaa levyjä?
  - joustava resurssien yhteiskäyttö
    - lue tiedosto levytä vai verkkopalvelimelta?
  - tiukka tietosuoja
    - kuka muu kuin Pekka tai Maija luki tietojani?

19.4.2010

Copyright Teemu Kerola 2010

35

## Käyttöjärjestelmä resurssien vartijana

- Suoritinaikaa reilusti kaikille
  - kukaan ei odota suoritinta ikuisesti
  - kriittiset prosessit saavat ajoissa suoritinaikaa
- Tiedostojen (koodi, data) tehokas käyttö
  - laitteesta ja sijainnista riippumaton käyttö
  - helppo yhteiskäyttö ja samalla tietojen suojaus
- Tietoliikenneverkkojen käyttö
  - laiteriippumaton käyttö
  - helppo yhteiskäyttö ja samalla tietojen suojaus
- Hallintokirjanpito

19.4.2010

Copyright Teemu Kerola 2010

36

## KJ järjestelmän eheyden turvaajana

- Varauduttu kaikkiin mahdollisiin virheisiin
- Sovellusohjelmat eivät voi häiritä KJ:tä tai muita prosesseja
  - tahallaan (esim. tietokonevirukset)
  - vahingossa (yleisin tapaus)
- Järjestelmä ei lukkiudu tai ”kaadu”
  - KJ:n omat tietorakenteet ovat aina eheitä
  - sovellusohjelmat eivät voi koskea KJ:n tietorakenteisiin
  - sovellusohjelmat aina lopulta antavat vuoron KJ:lle

19.4.2010

Copyright Teemu Kerola 2010

37

## Käyttöjärjestelmän rakenne

- Prosessien hallinta
  - prosessien luonti, tuhoaminen
  - prosessien välinen viestintä (IPC, Inter-Process Comm)
  - kenelle suoritinaikaa ja milloin?
- Muistin hallinta
  - miten keskusmuistia varataan eri prosessien käyttöön?
  - kunkin prosessin muistitilan hallinta
  - yhteiskäyttö ja tiedon suojaus
- Tiedostojen ja laitteiden hallinta
  - miten tiedostoja voidaan lukea/kirjoittaa?
  - yhteiskäyttö ja tiedon suojaus
- Verkon hallinta
  - miten kommunikoida muiden koneiden kanssa?

19.4.2010

Copyright Teemu Kerola 2010

38

## Käyttöjärjestelmän toteutus (kesk.)

- Joukko prosesseja ja/tai aliohjelmia
  - prosessit elävät omaa elämäänsä (root'ina?)
    - swapper (Unix) - muistinhallintaprosessi
    - init prosessi (Unix) - kaikkien käyttäjätason prosessien ”äiti”
    - laiteajurit
  - aliohjelmat suoritetaan sen hetkisen prosessin ympäristössä (etuoikeutetussa tilassa?)
    - keskeytyskäsittelijät
  - saavat kontrollin aina tarvittaessa
    - aliohjelmakutsut, SVC, viestit
    - ajastimet ja muut keskeytykset

19.4.2010

Copyright Teemu Kerola 2010

39

## KJ palvelun kontrollin palautus

Explisiittinen  
KJ-palvelun kutsu

- Aliohjelmakutsut
  - CALL → RETURN
- SVC
  - SVC → IRET
- Viestit
  - viesti → vastausviesti  
(lähettäjä odottaa vastausta RECEIVE:ssä)

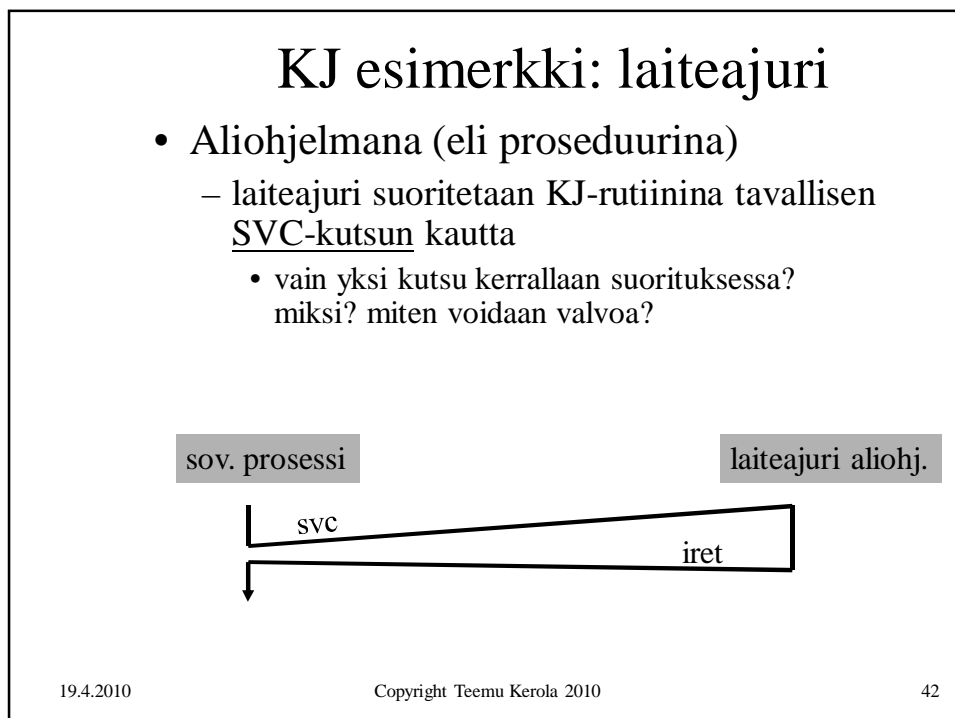
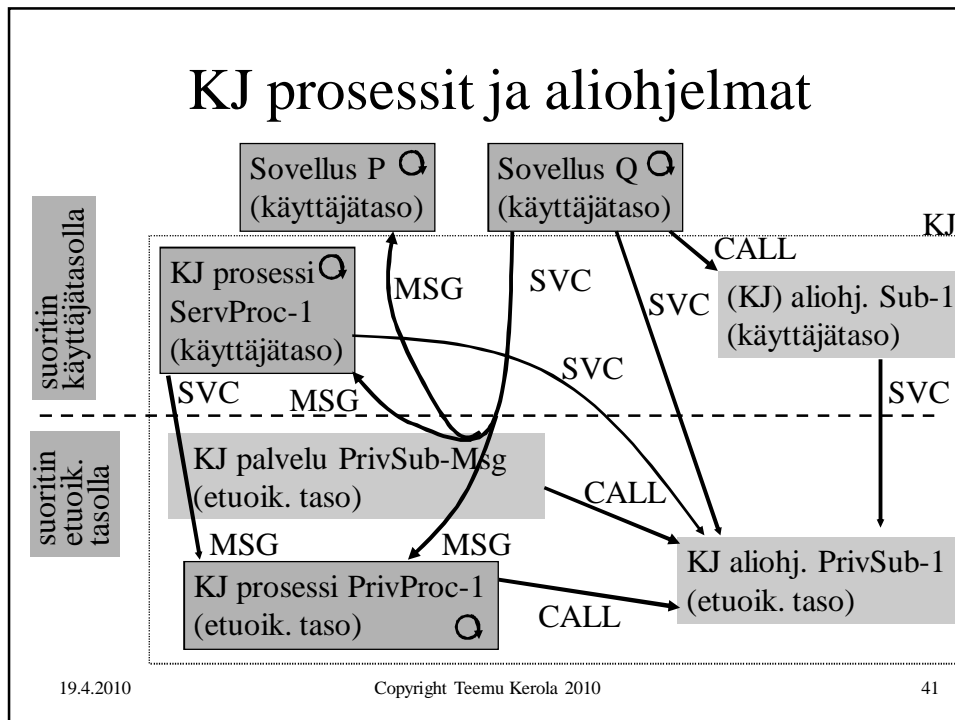
Implisiittinen  
KJ-palvelun kutsu

- Ajastimet ja muut keskeytykset
  - keskeytys → IRET

19.4.2010

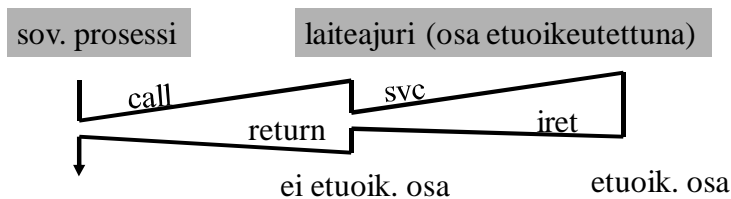
Copyright Teemu Kerola 2010

40



## KJ esimerkki: laiteajuri (jatk.)

- Aliohjelmana (eli proseduurina)
  - laiteajuri suoritetaan KJ-rutiinina tavallisen aliohjelmakutsun ja/tai SVC-kutsun kautta
    - osa tai kaikki koodista voi olla etuoikeutettua
    - vain yksi kutsu kerrallaan suorituksessa?  
miksi? miten voidaan valvoa?



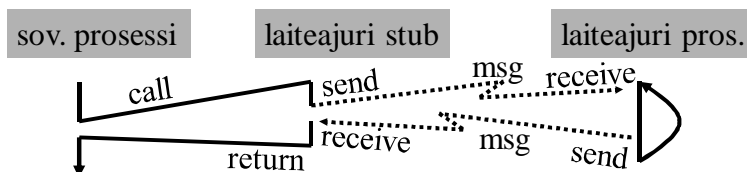
19.4.2010

Copyright Teemu Kerola 2010

43

## KJ esimerkki: laiteajuri

- Prosessina
  - proseduurina kutsuttu laiteajurin tynkä (stub) lähettää I/O-pyyynnön viestinä laiteajuriprosessille ja odottaa vastausta
    - tynkä voi olla käyttäjätilainen
    - ajuriprosessi voi olla (joskus) etuoikeutettu
    - vaatii prosessien välistä viestintää



19.4.2010

Copyright Teemu Kerola 2010

44

## -- Luennon 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 kurssi