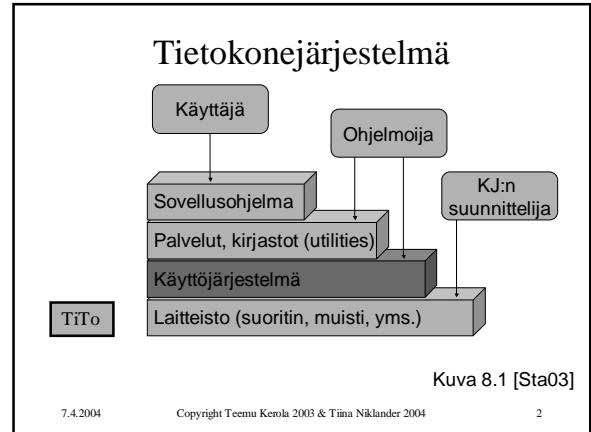


Luento 8

Ohjelman toteutus järjestelmässä

Prosessi
 Prosessin esitysmuoto
 järjestelmässä
 Käyttöjärjestelmä
 KJ-prosessit
 Käyttöjärjestelmien kehitys

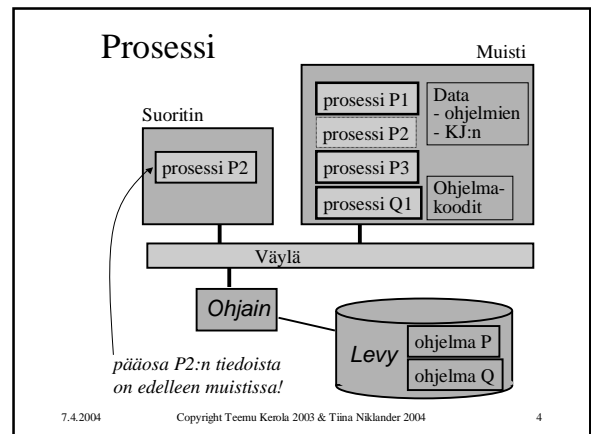
7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 1



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

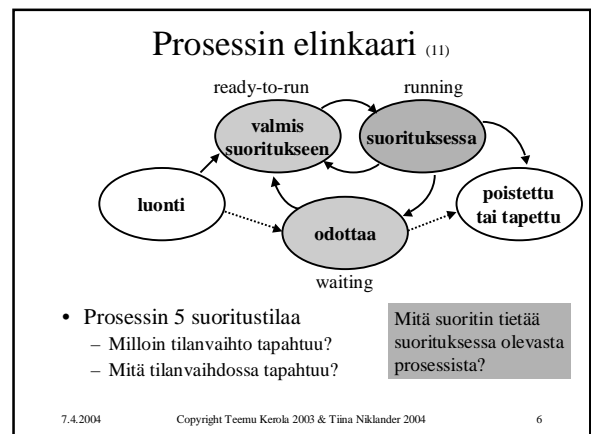
7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 3



Prosessin vaihto (3)

- 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?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 5



Prosessin esitysmuoto järjestelmässä

- PCB - Prosessin kuvaaja eli kontrollilohko (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
 - luodaan prosessin luonnin yhteydessä ja tuhoaan prosessin päättyessä
 - käyttöjärjestelmärutiinit manipuloivat PCB:tä

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 7

Prosessin kuvaajan sisältö ⁽⁹⁾

- Prosessin tunniste **14023**
- Prioriteetti **143** suorittimen vuoronantoa varten
- 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)

Ks. Minix esimerkin *struct proc* [Tane87], 1 kalvo

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 8

Prosessin tilanvaihdon toteutus ⁽⁵⁾

- Prosessin tilanvaihto tapahtuu siirtämällä prosessi (sen PCB) jonosta toiseen
 - ready-to-run jono (tai jonot) **odottaa suorittinta**
 - 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

Ks. Minix esimerkin *ready* [Tane87], 1 kalvo

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 9

Prosessit jonoissa ⁽¹⁾

waiting { Disk1, Timer, Msg from 1345 }

Running { 0188 }

R-to-R { 1056, 1766 }

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)

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 10

KJ esimerkki: I/O keskeytys ⁽⁵⁾

waiting { Disk1, Timer (S), Msg from Tdriver (U) }

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

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

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 11

KJ esimerkki: I/O keskeytys ^(ei anim)

waiting { Disk1 (Tdriver), Timer (S), Msg from Tdriver (U) }

R-to-R { R } → Q → Running { P }

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

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 12

KJ esim: aikaviipalekeskeytys (3)

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?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 13

KJ esim: aikaviipalekesk. (ei anim)

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?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 14

Prosessin vaihto (4)

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

Ks. Minix esimerkin `tty_int` [Tane87], kalvot INT 1-3/3

– usein keskellä prosessin vaihtoa suorittavaa KJ rutiinia

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 15

Prosessin prioriteetti (3)

- 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)

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 16

Käyttäjän näkökulma (käyttö)järjestelmään

- Miten järjestelmä toimii minun ohjelmani kanssa?
- Onko järjestelmä riittävän nopea pelaamaan suosikkipeleäni isolla näytöllä?
- Onko minun helppo asentaa uusi ohjelma koneelle?
- Onko minun helppo muuntaa (portata) ohjelmani tähän käyttöjärjestelmään?
- Miten muistin lisääminen vaikuttaisi minun ohjelmani nopeuteen?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 17

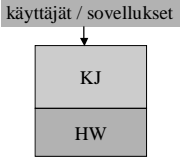
Käyttöjärjestelmän näkökulma (käyttö)järjestelmään

- Ovatko kaikki systeemin resurssit mahdollisimman hyvässä käytössä?
- Mikä on keskimääräinen jonon pituus (prosessien lukumäärä) suorittimelle?
- Minkä osan ajasta suoritin odottaa järkevää työtä?
- Minkä osan ajasta kovalevyn hakuvarsi on liikkeessä?
- Miten usein datamuistiviitteet löytyivät välimuistista?
- Miten muistin lisääminen vaikuttaisi nopeuteen?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 18

Käyttöjärjestelmä käyttöliittymänä laitteistoon

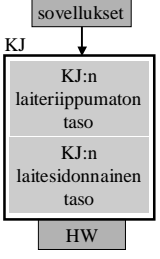
- Loppukäyttäjälle (ihmiselle)
- Sovellusohjelmille
- Piilottaa laitteiston erityispiirteet käyttäjiltä
 - käskykanta
 - konekäskyn rakenne
 - suorittimen toteutus ja suorittimien lukumäärä
 - I/O:n toteutus
 - I/O-laitteiden sijainti



7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 19

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



7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 20

Käyttöjärjestelmän tavoitteet (jatk)

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

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 21

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

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 22

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

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 23

Käyttöjärjestelmän rakenne (4)

- 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?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 24

Käyttöjärjestelmän toteutus (5)

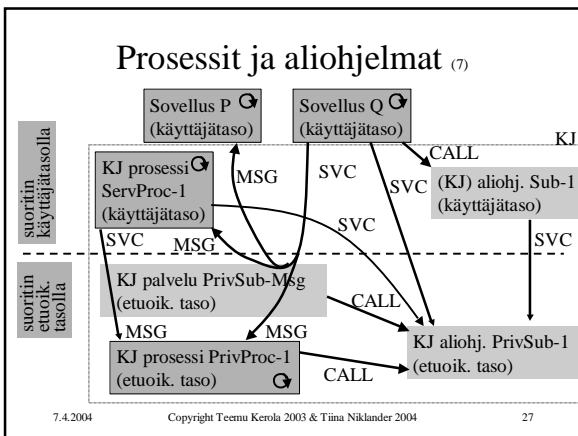
- Joukko prosesseja ja/tai aliohjelmaa
 - prosessit elävät omaa elämäänsä (etuoikeutetussa tilassa eli root'ina?)
 - swapper (Unix) - muistinhallintaprosessi
 - init prosessi (Unix) - kaikkien käyttäjätason prosessien "äiti"
 - laiteajurit
 - aliohjelmat suoritetaan sen hetkisen prosessin ympäristössä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

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 25

KJ palvelun kontrollin palautus

- Aliohjelmakutsut
 - CALL → RETURN
- SVC
 - SVC → IRET
- Viestit
 - viesti → vastausviesti (lähettäjä odottaa vastausta RECEIVE:ssä)
- Ajastimet ja muut keskeytykset
 - keskeytys → IRET

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 26



KJ esimerkki: laiteajuri

- Aliohjelmana (eli proseduurina)
 - laiteajuri suoritetaan KJ-rutiinina tavallisen SVC-kutsun kautta
 - vain yksi kutsu kerrallaan suorituksessa? miksi? miten voidaan valvoa?

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 28

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?

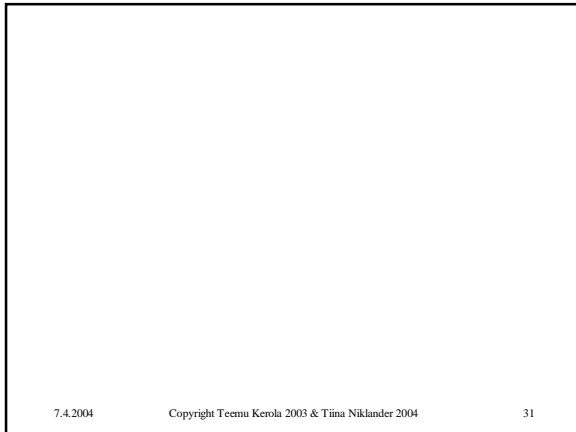
7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 29

KJ esimerkki: laiteajuri

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

Ks. Minix esimerkki floppy disk driver [Tane87], 2 kalvoa

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 30




Käyttöjärjestelmien kehitys

- Varhaiset järjestelmät
 - ei käyttöjärjestelmää, kytkimiä
- Eräajojärjestelmät (batch systems)
 - osaa suorittaa yhden työn kerrallaan
 - ja jatkaa sitten vuorossa seuraavalla
- Moniajojärjestelmät (multiprogramming)
 - vuorottaa useaa samaan aikaan suorituksessa olevaa työtä
 - edelleen eräajoa
- Osituskäyttöjärjestelmät (timesharing)
 - suorakäyttö (on-line), useita samanaikaisia käyttäjiä

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 32

Varhaiset järjestelmät

- Ei käyttöjärjestelmää, kytkimiä
- Vain yksi ohjelma kerrallaan
 - ladataan suoritukseen konekielisenä
 - suoritetaan
 - tulokset tallennetaan
- Käyttäjä ei välttämättä ollut suoraan yhteydessä koneen kanssa
- Suoritusvuorojen jakelu
 - Varauslista paperilla (aikajaksot)
 - Kehittyneempi menettely:
 - Operaattori, jolle toimitettiin suoritettavat ohjelmat



7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 33


Eräajojärjestelmät (batch systems)

- Vain yksi ohjelma kerrallaan suorituksessa,
- mutta tietokoneella pieni ohjelmanpätkä (lataaja, monitori), joka osaa 'automaattisesti' ladata työn ja suorittaa sen
- Operaattori järjestää työt jonoon
 - laittaa korttipakat kortinlukijaan tai
 - lataa magneettinauhat nauhuriin
- Käyttäjä toimittaa ohjelmansa suoraan korttipakkana tai magneettinauhana operaattorille ja saa joskus suorituksen jälkeen tuloksen paperilla tai magneettinauhalla

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 34

Moniajojärjestelmät (multiprogramming)

- Edelleen operaattorien suorittamaa eräajoa
- Nyt samanaikaisesti suorituksessa useita töitä
 - kun yksi työ odottaa oheislaitetta, suoritin voi
 - suorittaa sillä aikaa jotain toista ohjelmaa
- Tämä toimintaperiaate käytössä nykyisissäkin käyttöjärjestelmissä (ns. tausta-ajo)
 - prosessi ei ole suorittimella, jos se odottaa oheislaitteelle annetun tehtävän valmistumista




IBM System/360

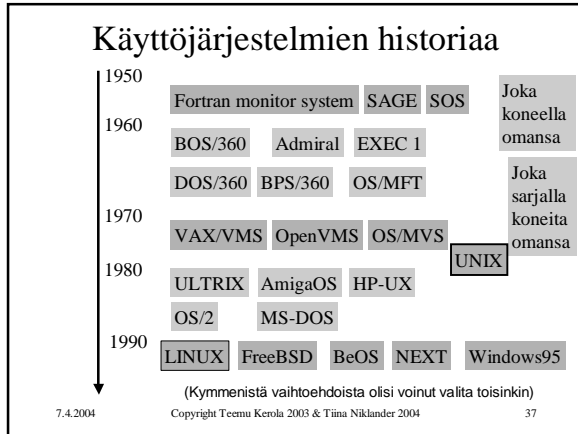
7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 35

Osituskäyttöjärjestelmät (timesharing)

- Suorakäyttö (on-line) – käyttäjällä on suora yhteystietokoneeseen
- useita samanaikaisia käyttäjiä
- Valtaosa nykyisistä käyttöjärjestelmistä tarjoaa käyttäjälle ajantasaisen yhteyden tietokoneeseen (esim. kaikki PC:t)



7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 36



-- 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 kurssit, RIO, Hajjärj

7.4.2004 Copyright Teemu Kerola 2003 & Tiina Niklander 2004 38