LUT
University

CT30A3370 - KÄYTTÖJÄRJESTELMÄT JA SYSTEEMIOHJELMOINTI 6 OP

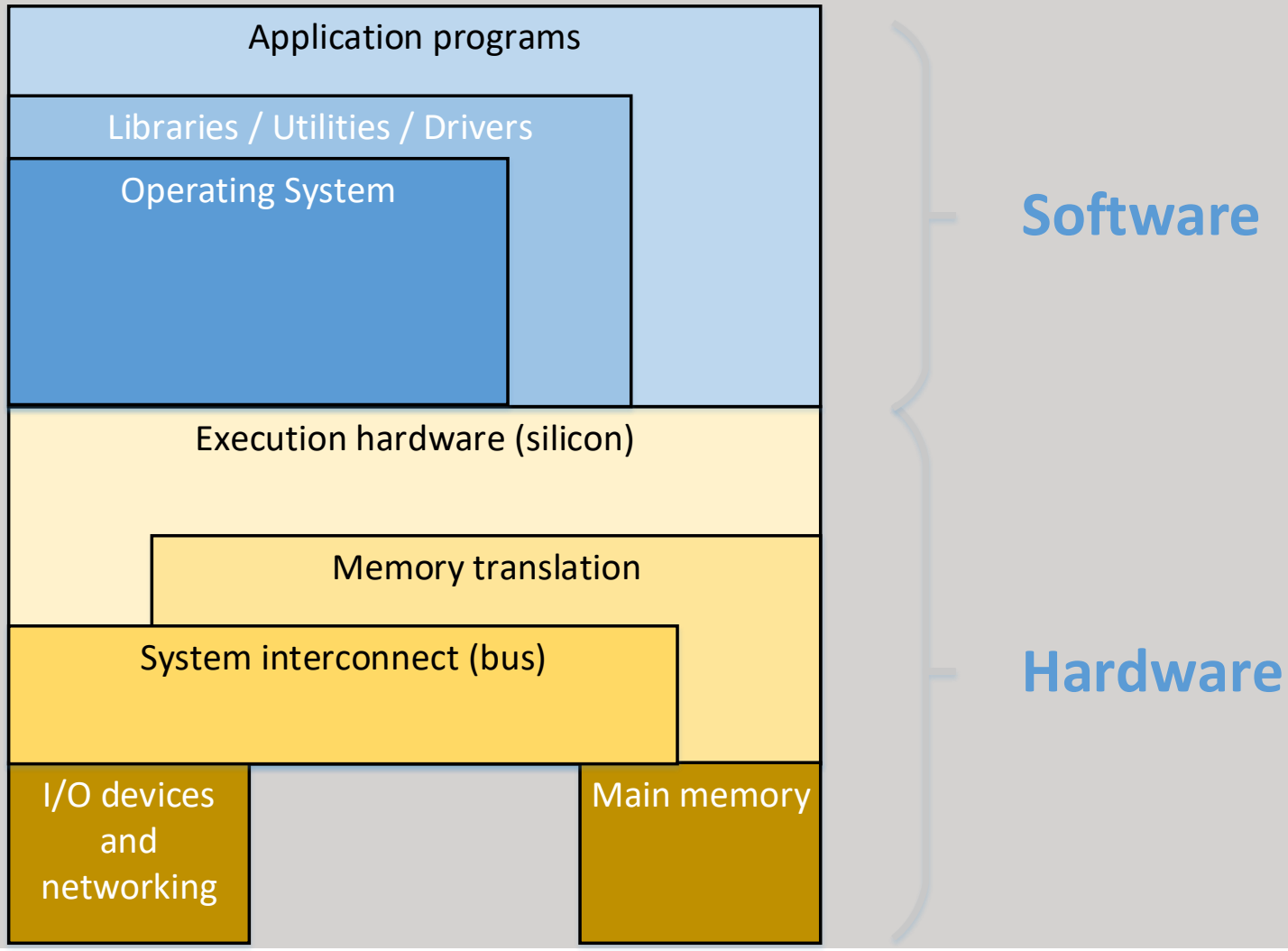
Jussi Kasurinen (etu.suku@lut.fi)

Osa kalvoista Timo Hynnisen 2016 materiaaleista

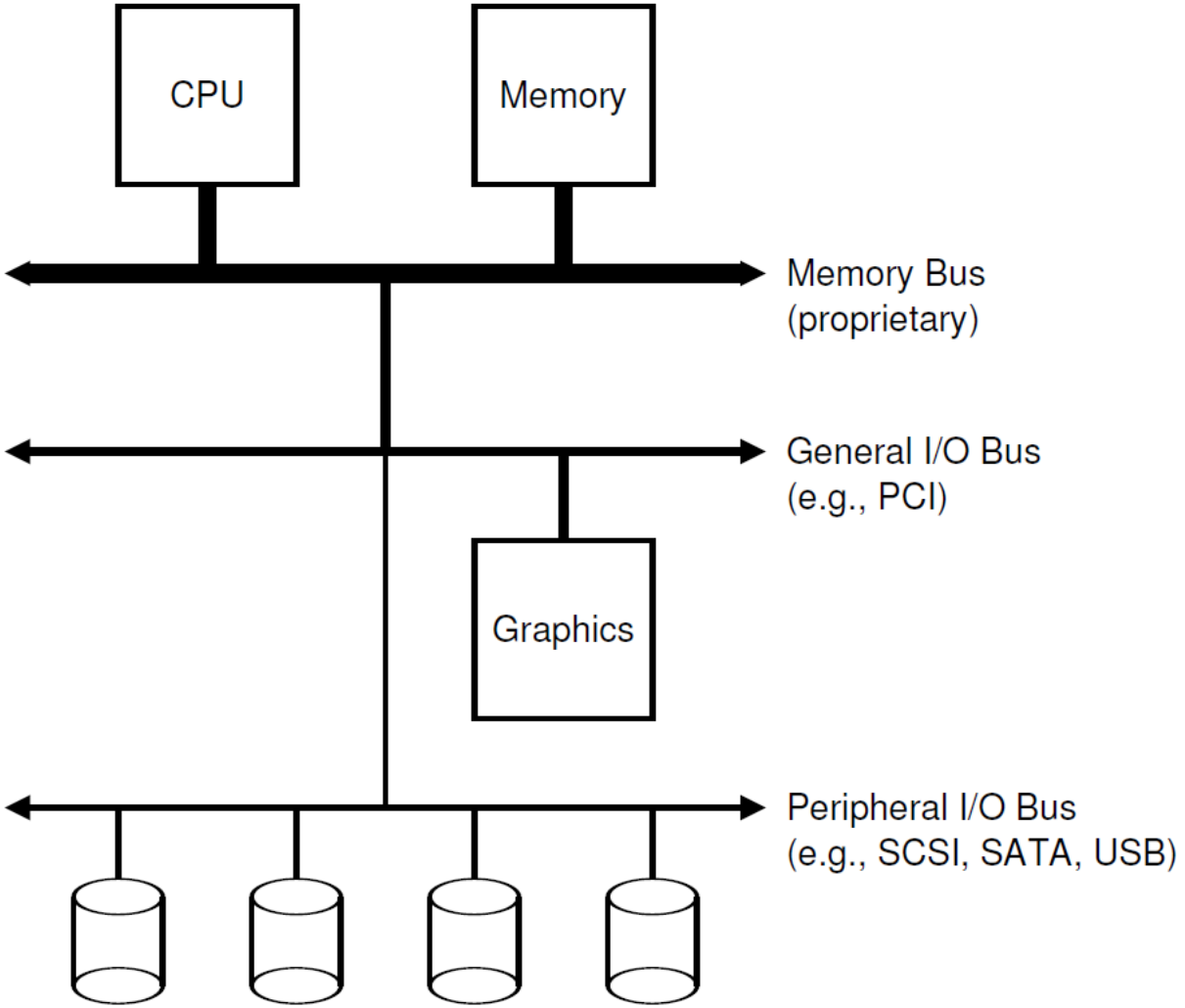
EDELLISILTÄ LUENNOILTA:

CT30A3370 - Käyttöjärjestelmät ja systeemiohjelmointi

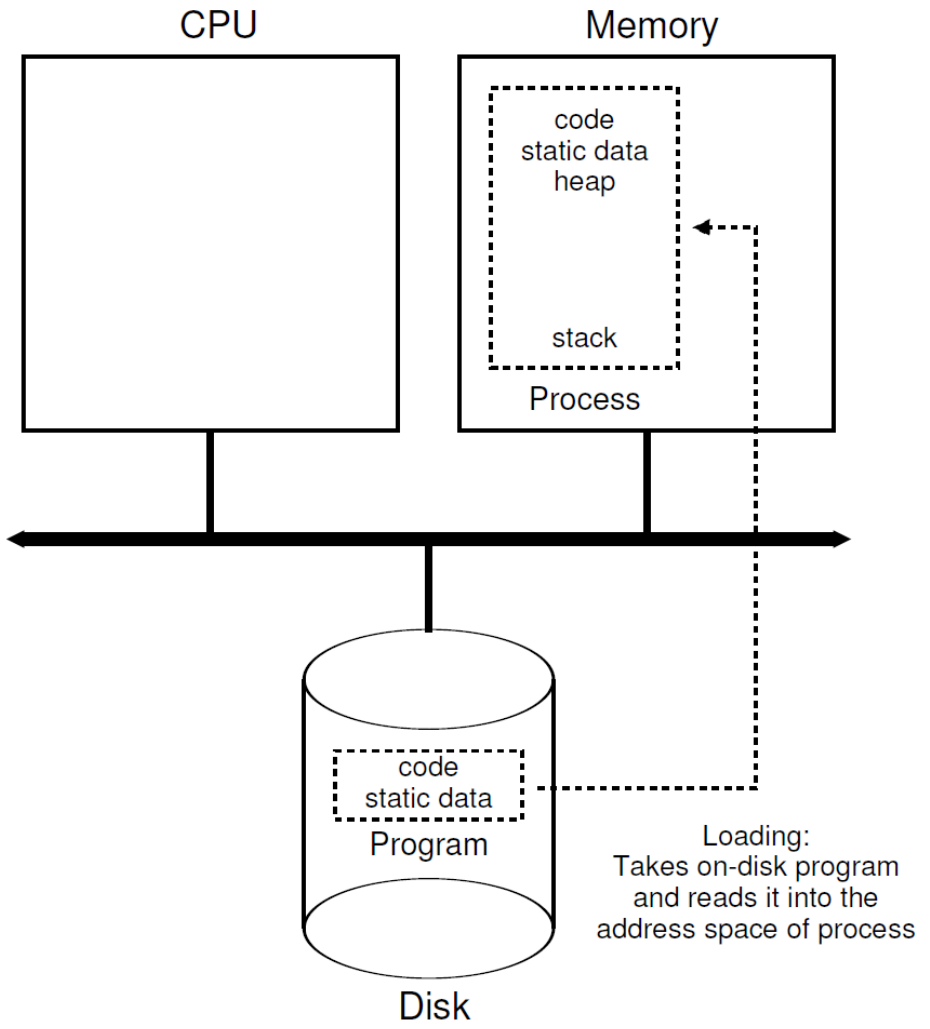
MODERNI TIETOKONE



NYKYINEN (PC) TIETOKONE PERIAATTEELLISELLA TASOLLA



OHJELMA VS. PROSESSI



PROSESSINOHJAUS

CT30A3370 - Käyttöjärjestelmät ja systeemiohjelmointi

RE: PROSESSIT

- Käyttöjärjestelmän ensimmäinen ja tärkein tehtävä oli siis koordinoida kaikkea käskyjen suoritusta, mitä tietokoneessa tapahtuu.
- Tämä on oikeassa elämässä vaikeaa
  - Meillä on useita käyttöjä, useita ohjelmia, useita laitteita jne.
- Millä tällaista kaaosta hallitaan
  - Vastaus: Pilkkomalla monimutkaiset ongelmat pienempiin osiin
  - Kaikkien ongelmien kanssa ei tarvitse taistella samanaikaisesti
  - Kun meillä on n kappaletta pienempiä ongelmia, niin ratkotaan niitä yksi kerrallaan.

PROSESSIT

- Prosessissa on kaksi osaa:
  - Peräkkäinen suoritus
    - Prosessissa ei ole mitään rinnakkaisuutta
    - kaikki suoritus on peräkkäistä, yksi käsky toisensa jälkeen
  - Prosessin tila
    - Kaikki muisti (muistipaikat), joihin prosessin suoritus vaikuttaa
      - rekisterit, keskusmuisti, tiedostot...

VIRTUALISOINTI

- Lyhyesti: ohjelmat saavat oman virtuaalisen prosessorin ja muistin.
- Ohjelmat toimivat omassa kehässään kiltisti tietämättä tai välittämättä siitä, että eivät ole ainoa käynnissä oleva prosessi.
- Käyttöjärjestelmän hommaksi jää huolehtia, että fyysiset resurssit jaetaan siten että virtualisointi toimii oikein.
- …Ja tarjota API (application protocol interface) että ohjelmat ja laitteet pääsevät resursseihin käsiksi.

VIRTUALISOINTI

- Kun puhutaan prosesseista käyttöjärjestelmän tasolla…
  - Kun puhutaan samanaikaisuudesta ja moniajosta, puhutaan oikeasti prosessien säikeistä ja niiden synkronoinnista
  - Kun puhutaan ohjelmien suorituksen turvallisuudesta tarkoitetaan nimenomaan prosessien muistiavaruuksia.

MONIAJO JA AJOITUS

- Jokainen prosessi mitä suoritetaan kuvittelee tai “näkee” asiat siten, kuin niillä jokaisella olisi oma suoritin.
- Oikeasti suorittimia on vähemmän kuin prosesseja
  - Usein kaikki prosessit jakavat saman suorittimen laskentatehoa
- Vuorontaja, scheduler, on se osa käyttöjärjestelmää, joka päättää, mitä prosessia kulloinkin suoritetaan

MONIAJO JA AJOITUS

- Miten prosessi tai säie loppupelissä suoritetaan?
  - Ladataan suorittimelle muistiin sen tila (eli rekistereihin, käskyosoitinrekisteriin ja käskyosoittimeen) ja hypätään ohjelman suorituksen alkuun.
  - Kuulostaa yksikertaiselta ja niin se onkin.

Ohjelman käynnistäminen ja suorittaminen

OS	Program
Create entry for process list	
Allocate memory for program	
Load program into memory	
Set up stack with argc/argv	
Clear registers	
Execute **call** main()	
	Run main()
	Execute **return** from main
Free memory of process	
Remove from process list	

MONIAJO JA AJOITUS

- Entä, jos halutaan keskeyttää prosessi tai säie?
  - Tämähän oli koko moniajon idea: Ei jäädä odottamaan yhden prosessin valmistumista, vaan pidetään monta ajossa samanaikaisesti
  - Tai prosessi on muuten vaan odottavassa tilassa, esimerkiksi odottaa I/O-laitetta
  - Vuorontajan on saatava jotenkin suorittimen kontrolli takaisin!

MONIAJO JA AJOITUS

- Kun vuorontaja saa järjestelmän (suorittimen) hallinnan takaisin, puhutaan keskeytyksestä
- Keskeytykset voivat olla
  - Aikaperusteisia
  - Tapahtumaperusteisia
    - I/O-laitteen tilassa tapahtuu muutos, esim. levylle kirjoittaminen on saatu valmiiksi, näppäimistöllä painetaan näppäintä..
- Vuorontaja pitää kirjaa prosesseista, jotka ovat parhaillaan suoritettavina

MONIAJO JA AJOITUS

Time	$Process_0$	$Process_1$	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	$Process_0$ initiates I/O
4	Blocked	Running	$Process_0$ is blocked,
5	Blocked	Running	so $Process_1$ runs
6	Blocked	Running	
7	Ready	Running	I/O done
8	Ready	Running	$Process_1$ now done
9	Running	–	
10	Running	–	$Process_0$ now done

VUORONTAJA

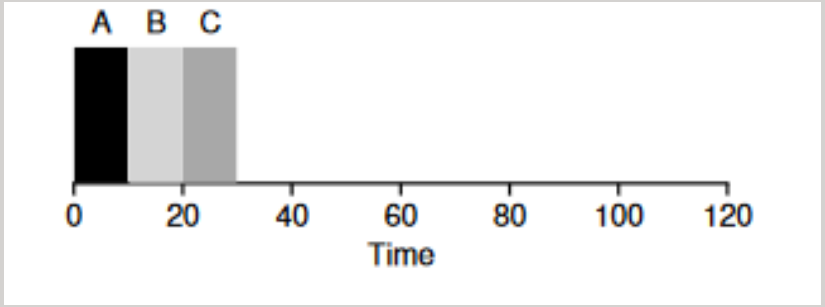
- Miten se valitsee suoritettavan prosessin?!
- Jos prosesseja on 0 tai 1 homma on helppoa (ei tehdä mitään tai suoritetaan yksi prosessi)
- Jos 2 tai enemmän.. Tarvitaan joku vuorotusalgoritmi prosesseille! Eli pitää jotenkin laittaa prosessit tärkeysjärjestykseen...

VUORONTAJA

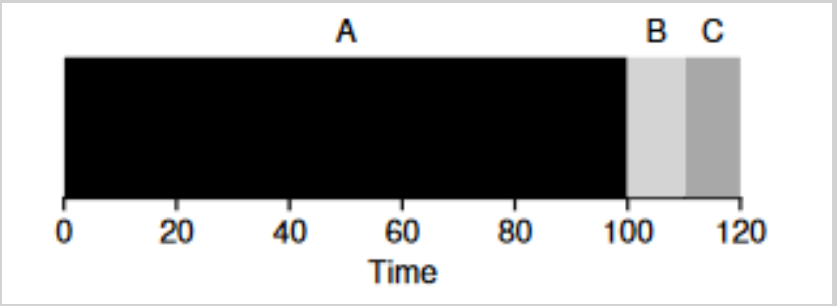
- LIFO
- FIFO
- First-Come-First-Served FCFS
- Round Robin RR
- Shortest Process Next SPN
- Shortest Remaining Time SRT
- Highest Response Ratio Next HRRN (minimoi läpimenoaikaa, valitsee seuraavaksi sen, joka on odottanut kauimmin.)

FIFO

- A, B, C tässä järjestyksessä, kaikki 10 sekunnin mittaisia.

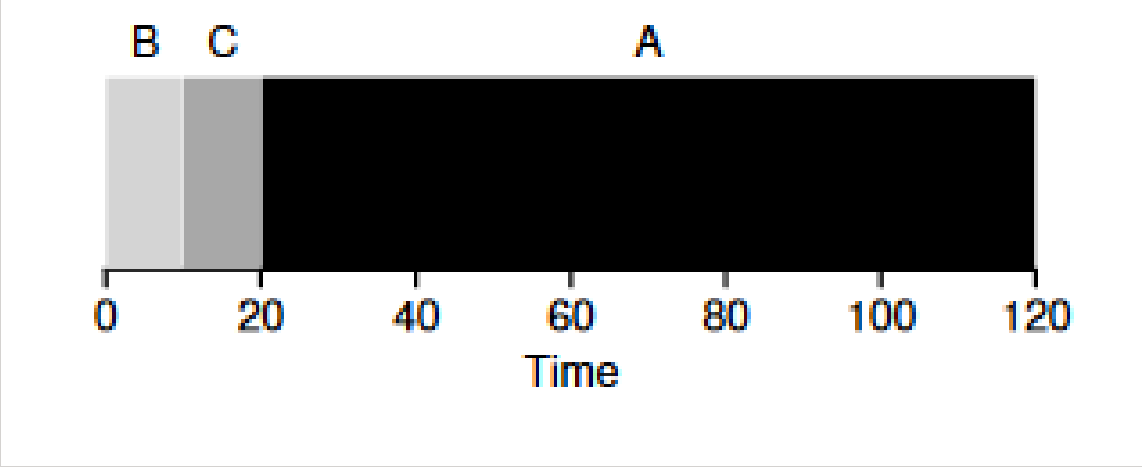


- Jos A olisi paljon pidempi, B ja C jää jumiin.



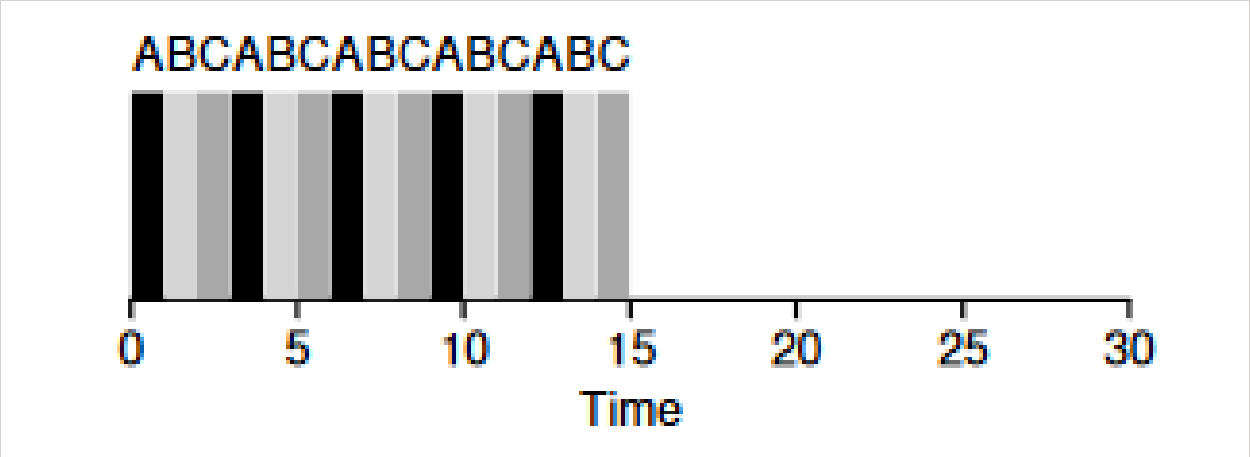
SJF – SHORTEST JOB FIRST /PROCESS NEXT

- A, B, C saapuu samaan aikaan, A 100 muut 10.
  - Miksi tämäkään ei toimi?



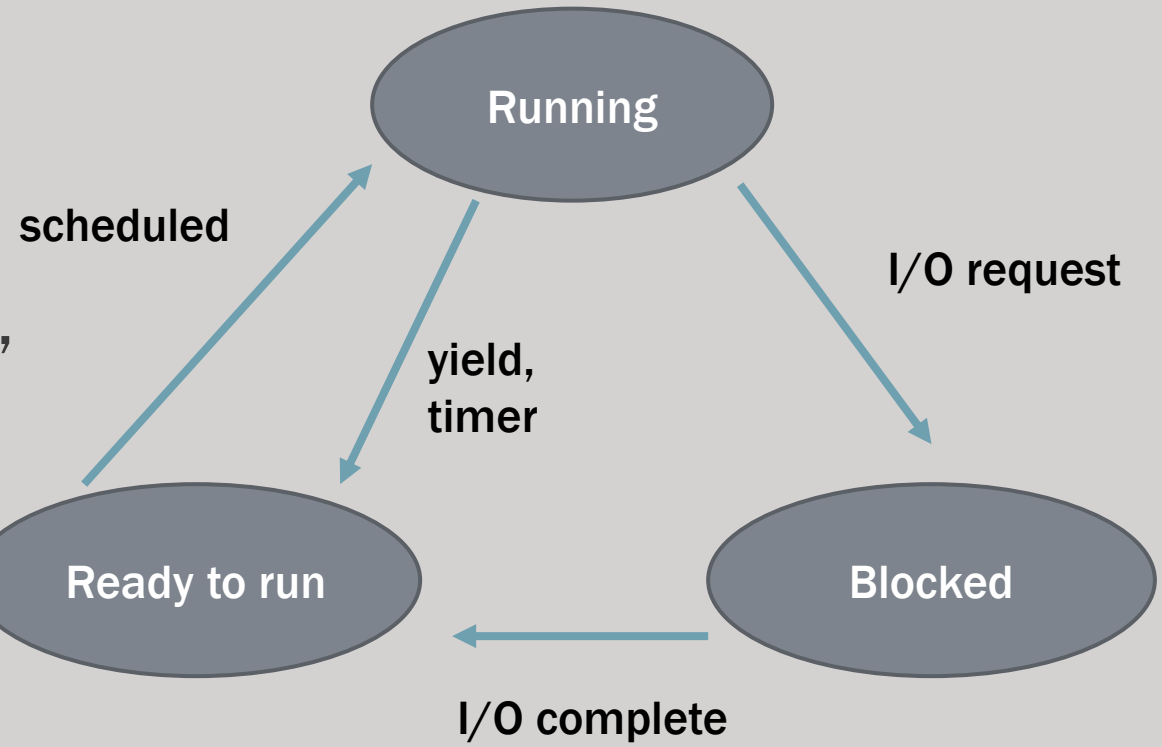
ROUND ROBIN – HYVÄ VASTEELLE, HUONO TEHOLLE

- A, B, C sisällä, kaikki 5 mittaisia.
  - Vaihtoväli 1.
  - Miksi tämä ei toimi aina?



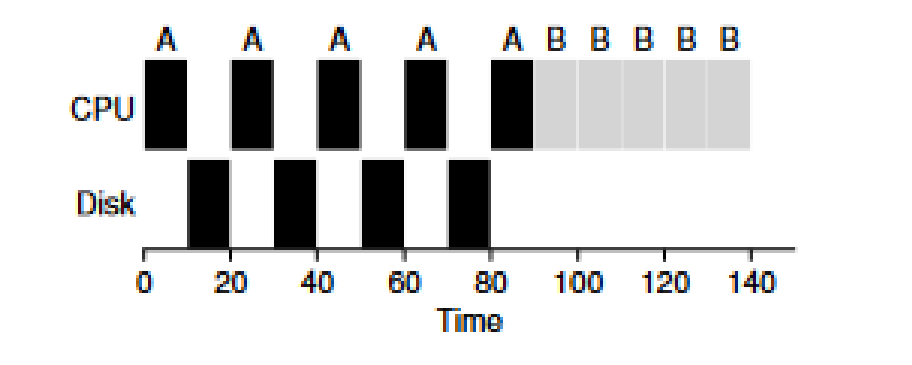
PROSESSI VOI OLLA KOLMESSA TILASSA:

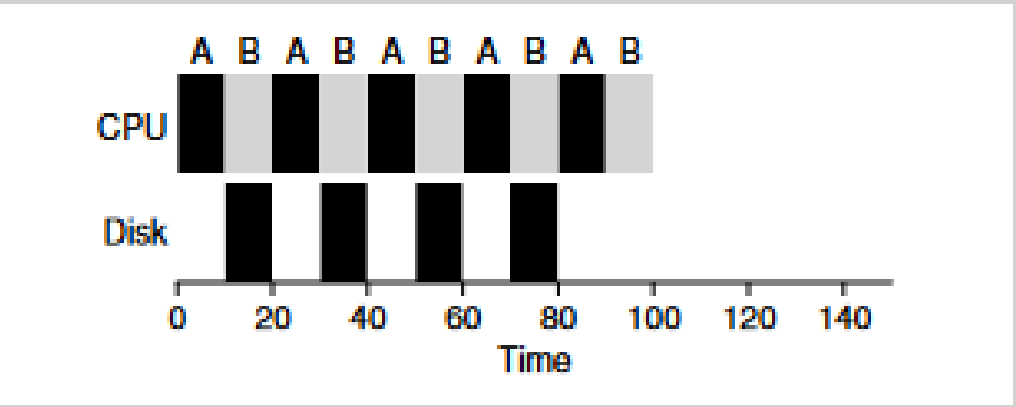
- Running
  - Suorituksessa suorittimella nyt
- Blocked
  - Odottaa jotain muuta resurssia, vaikkapa I/O-laitetta
- Ready to run
  - Vuorontajan listalla, odottamassa suoritusta



I/O-OPERAATIOT

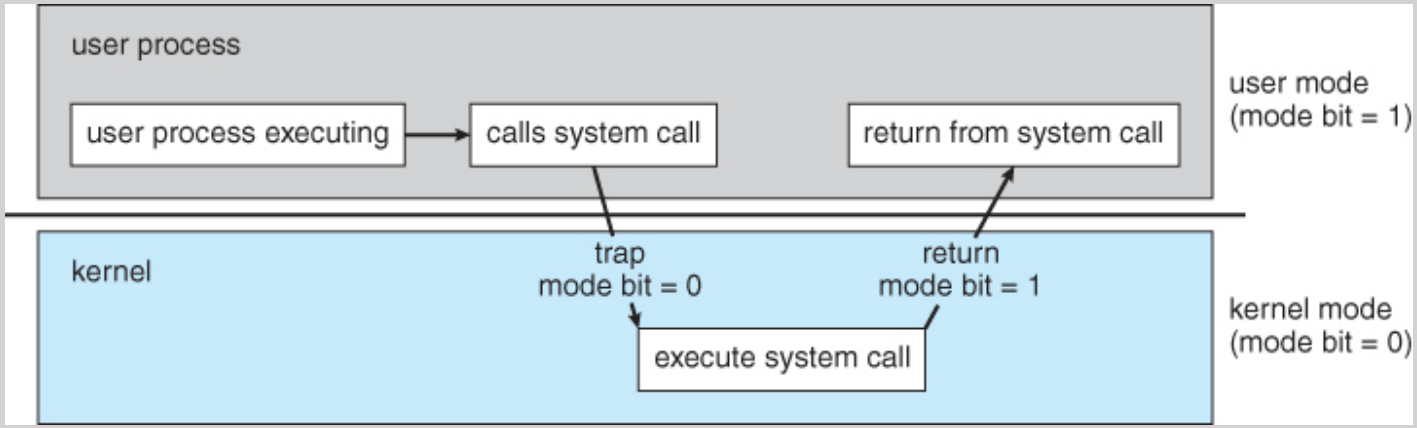
- Esim. levylle kirjoittaminen on jotain minkä fiksu vuorontaja huomioi!



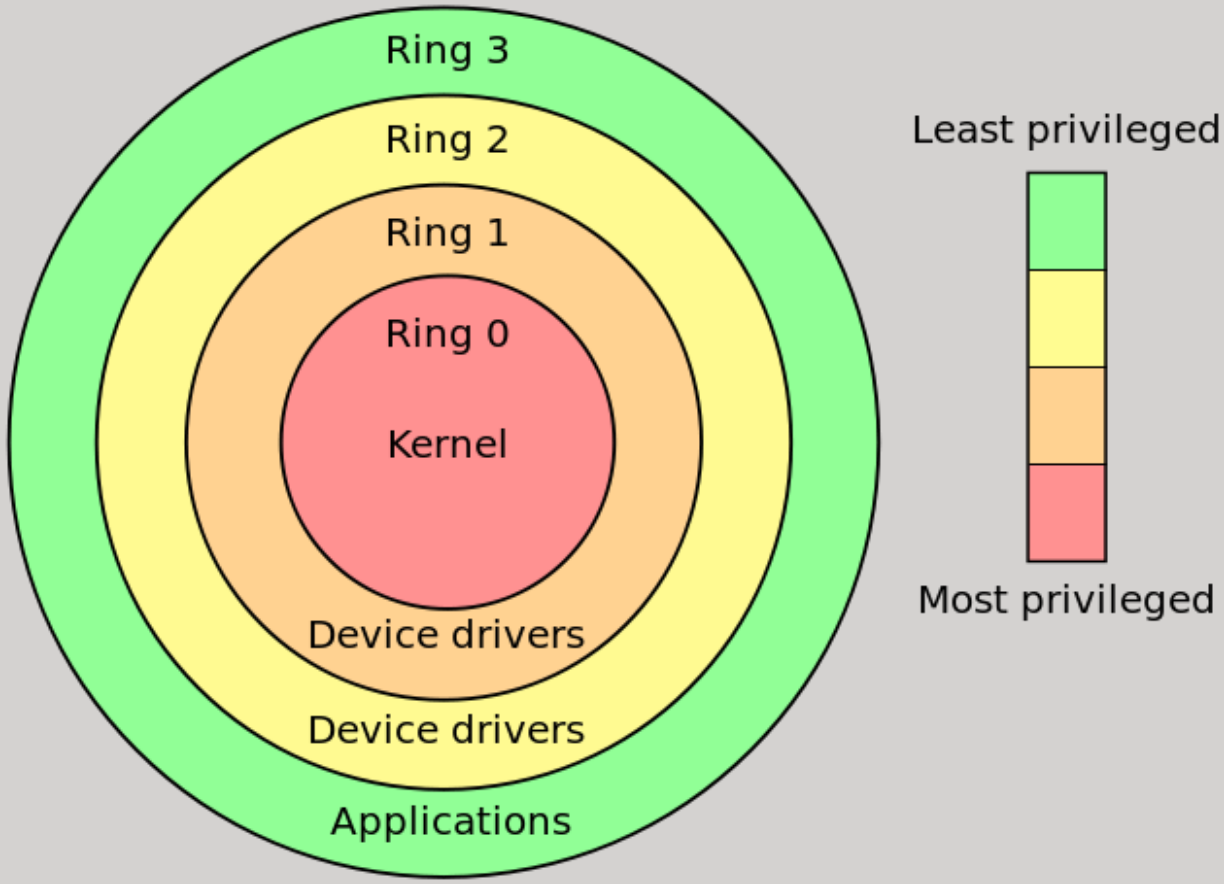


VÄLIHUOMIO; MITÄ TAPAHTUU KUN TULEE KESKEYTYS?

- Lyhyesti; prosessi menettää vuoronsa koska se tarvitsee käyttöjärjestelmältä apua, tai se luovuttaa vuoronsa pois (vaikkapa odottaessaan toista prosessia), tai sen vuoro päättyy.
  - Esimerkiksi prosessi pyytää levyltä tietoa tai haluaa tallentaa jotain; laitteiden suora käyttö pitää tehdä kernel-modessa (unixissa) ja sitä valvoo käyttöjärjestelmä, ei ajossa oleva prosessi.



RE: X86-PROSESSIEN PRIVILEEGIOTASOT



Prosessin vaihto aikakeskeytyksellä

OS @ boot (kernel mode)	Hardware	Program (user mode)
initialize trap table		
	remember addresses of...	
	syscall handler	
	timer handler	
start interrupt timer		
	start timer	
	interrupt CPU in X ms	

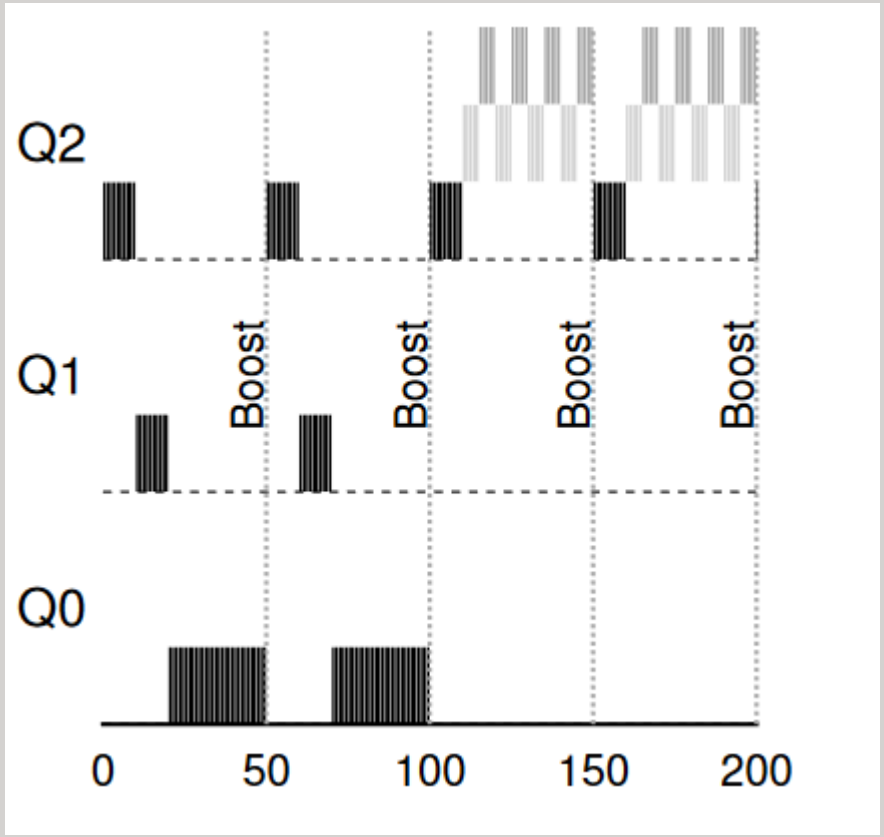
OS @ run (kernel mode)	Hardware	Program (user mode)
		Process A
		...
	timer interrupt	
	save regs(A) to k-stack(A)	
	move to kernel mode	
	jump to trap handler	
Handle the trap		
Call `switch()` routine		
save regs(A) to proc-struct(A)		
restore regs(B) from proc-struct(B)		
switch to k-stack(B)		
return-from-trap (into B)		
	restore regs(B) from k-stack(B)	
	move to user mode	
	jump to B's PC	
		Process B
		...

MULTI-LEVEL FEEDBACK QUEUE (SE MITÄ BSD LINUX JA WINDOWS KÄYTTÄÄ)

- Monitasoinen, prioriteettijonot
- Ylimmän prioriteetin ensin
- Samalla prioriteetilla round robinia
- Jos prosessi käyttää koko vuoronsa, se menettää yhden prioriteettitason.
- Kaikki aloittaa ylimmällä prioriteetilla
  - Kaikki siirretään tietyin väliajoin ylimpään prioriteettiin.
- Matala prioriteetti saa olla kerralla pidempään vuorossa.

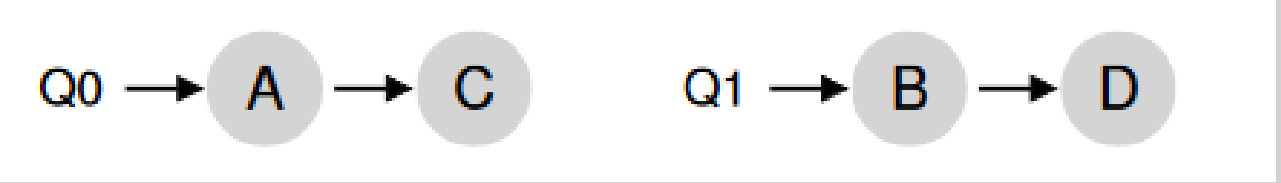
MULTI-LEVEL FEEDBACK QUEUE (SE MITÄ BSD LINUX JA WINDOWS KÄYTTÄÄ)

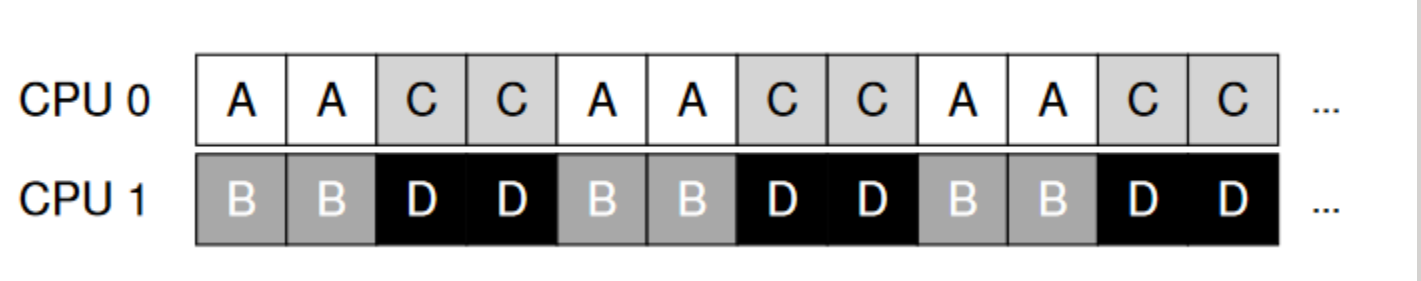
- Interaktiivisuus pysyy korkeana.
  - Lyhyet asiat tapahtuvat nopeasti.
- Laskentaintensiiviset prosessit vajoaa alaspäin “taustalle” ja niitä suoritetaan aina kun ei ole kiireellisempää tiellä.
  - Nostoaika varmistaa että kaikki kuitenkin saa välillä prosessoriaikaa.



VUORONNUS MONELLE PROSESSORILLE?

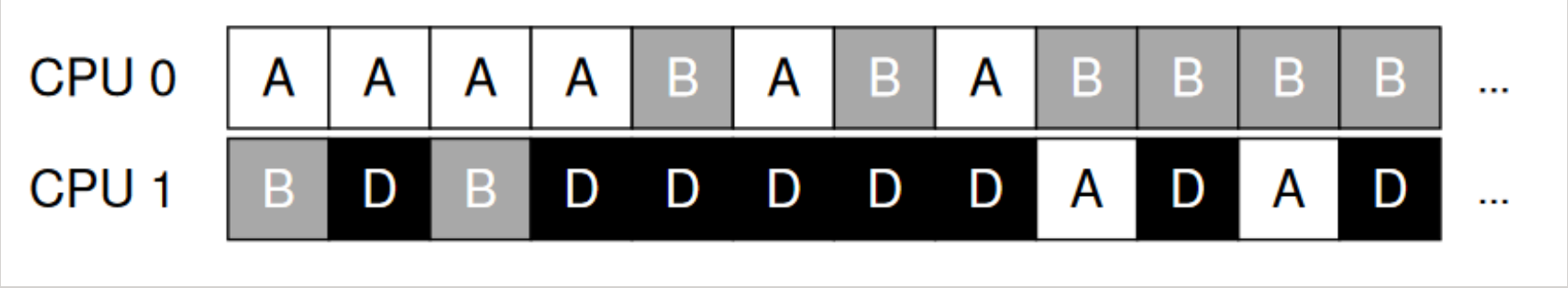
- Jos suorittimia on useita, päättää vuorontaja siitä, mihin suoritinjonoon mikäkin prosessi menee.





VUORONNUS MONELLE PROSESSORILLE?

- Jos prosessoreilla on epätasainen kuorma, voi vuorontaja myös siirrellä prosesseja jonosta toiseen.
  - Tämän toteutukseen ja siirtyjän valintaan on joka käyttiksellä oma ratkaisutapansa.



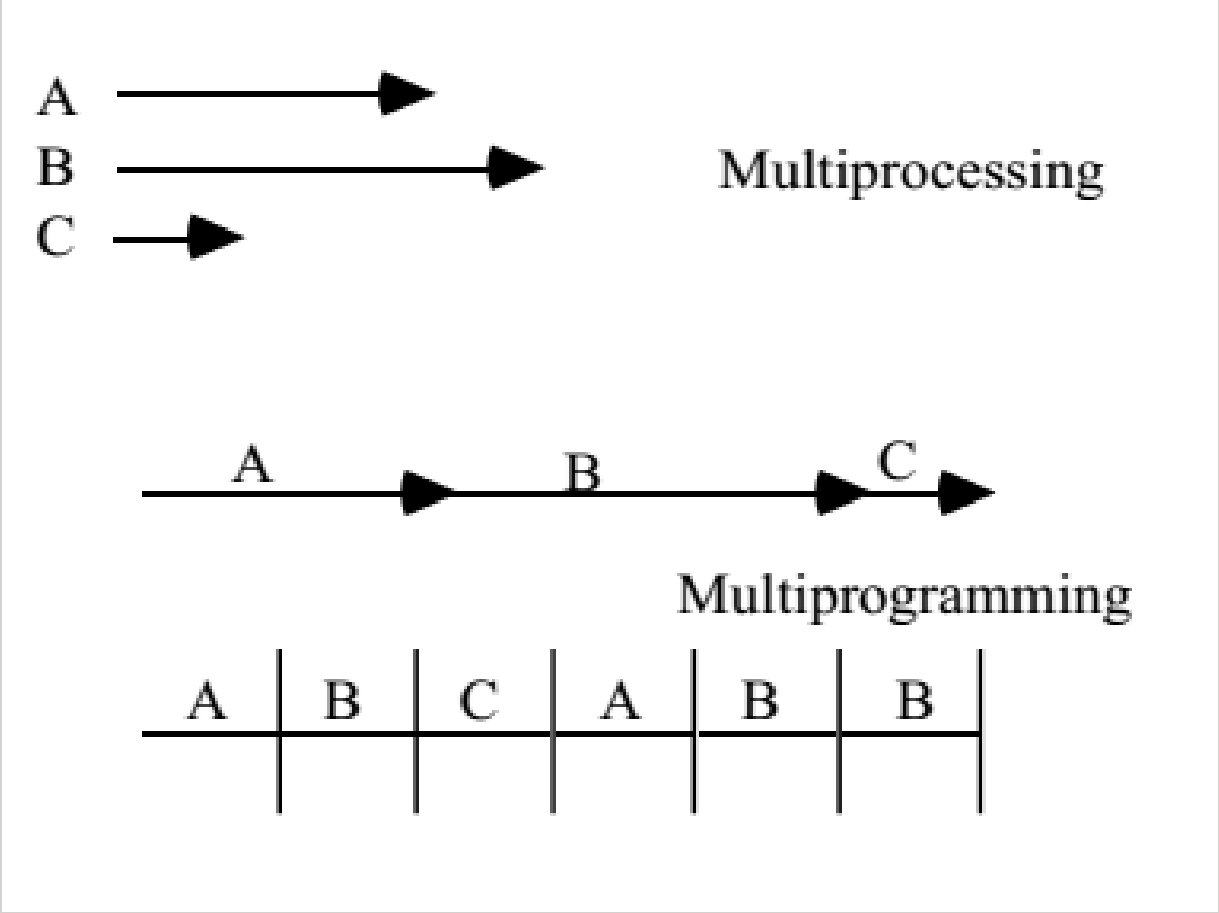
MONIAJOSTA JA VUORONTAJASTA

- Mikä tässä sitten on niin vaikeaa?
  - Vuorontaja tekee itsenäisiä päätöksiä, sitä ei voi ohjata suoraan.
  - Mistään ei voi päätellä tai tietää etukäteen, missä järjestyksessä prosessit tai niiden osat menee suorittimelle!
    - Ei edes silloin kun kyseessä on saman ohjelman säikeet tai prosessit!
  - Kaikille pitää antaa vuoro, että ne eivät näänny.
    - Toisaalta jatkuva vuoronvaihto tuhlaa laskentatehoa.
  - Vuorontaja ei myöskään vähääkään välitä siitä, oliko prosessilla joku oma juttu kesken.

MONIAJOSTA

- Yhdellä vai useammalla suorittimella?
  - Pidetään mielessä Moniajon toteutuksen vaihtoehdot
  - Voidaanko oikeasti harrastaa samanaikaista laskentaa, vai joudutaanko vuorottamaan prosesseja?
- Multiprocessing = usean suorittimen moniajo
- Multiprogramming = yhden suorittimen moniajo (prosessien moniajo)

MONIAJOSTA



MONIAJOSTA

- Toki toteutuksissa ei sinänsä ole eroa, usean suorittimen tapauksessa kaikki on paremmin, koska prosessit saadaan suoritettua samanaikaisesti eli nopeammin!
  - Vai onko tässä kuitenkin joku ongelma?
  - Jos vuorontaja voi valita kummasta tahansa tavasta suorittaa prosesseja, ohjelmien tulee ottaa se huomioon!
  - Mitä, jos prosessit esimerkiksi jakavat resursseja? Tai kommunikoivat keskenään?
  - Jos yksi prosessi muokkaa resurssia joka on toisella käytössä… voi syntyä ongelmia

DISFUNKTIONAALISET KÄMPPIKSET

- Kellonajat on ulkomaanformaatissa
  - Mutta henkilökohtaisesti minusta tarina on niin tyhmä, että saattaahan tämä tapahtua myös kolmelta aamuyöstä..

	Person A	Person B
3:00	Look in fridge. Out of milk.	
3:05	Leave for store.	
3:10	Arrive at store.	Look in fridge. Out of milk.
3:15	Buy milk.	Leave for store.
3:20	Arrive home, put milk away.	Arrive at store.
3:25		Buy milk.
3:30		
		Arrive home, put milk away.
		Oh no!

MAITOKRIISI

- Miten estää tällainen maitokriisi?
- Ongelman reunaehdot:
  - Vain yksi henkilö kerrallaan ostaa maitoa
  - Kuka tahansa voi ostaa maitoa tarvittaessa
- Mitäs jos jätetään viesti: jääkaapin oveen lappu, et “olen kaupassa ostamassa maitoa”
  - Siis eräänlainen lukitusmekanismi
- Laita lappu jääkaapin oveen (lukitus)
- Ota muistilappu pois jääkaapin ovesta (lukituksen poistaminen)
- Jos ovessa on lappu, älä lähde ostamaan maitoa!

MAITOKRIISI

```
if (noMilk) {
    if (noNote){
        leave Note;
        buy milk;
        remove note;
    }
}
```

MAITOKRIISI

- Miksi tämä ei toimi?
- Mietitäänpä, mitä tapahtuu prosessien vuorottamisessa
  - Mitä, jos prosessin suoritus keskeytetään sen jälkeen, kun on tarkastettu onko maitoa, ja onko lukkoa, mutta ennen lukon asettamista ja maidon ostamista!
- Tämä ratkaisu itse asiassa vain pahentaa asiaa!
  - Suoritus epäonnistuu satunnaisesti, vain joskus
  - Tällaista vikaa on aika hankala alkaa debuggaamaan...

MAITOKRIISI

- Ratkaisu 2:
  - Okei, mitäpä jos laitetaan muistilapuille nimet
  - “Henkilön / prosessin A post-it lappu ja Henkilön / prosessin B muistilappu” erikseen

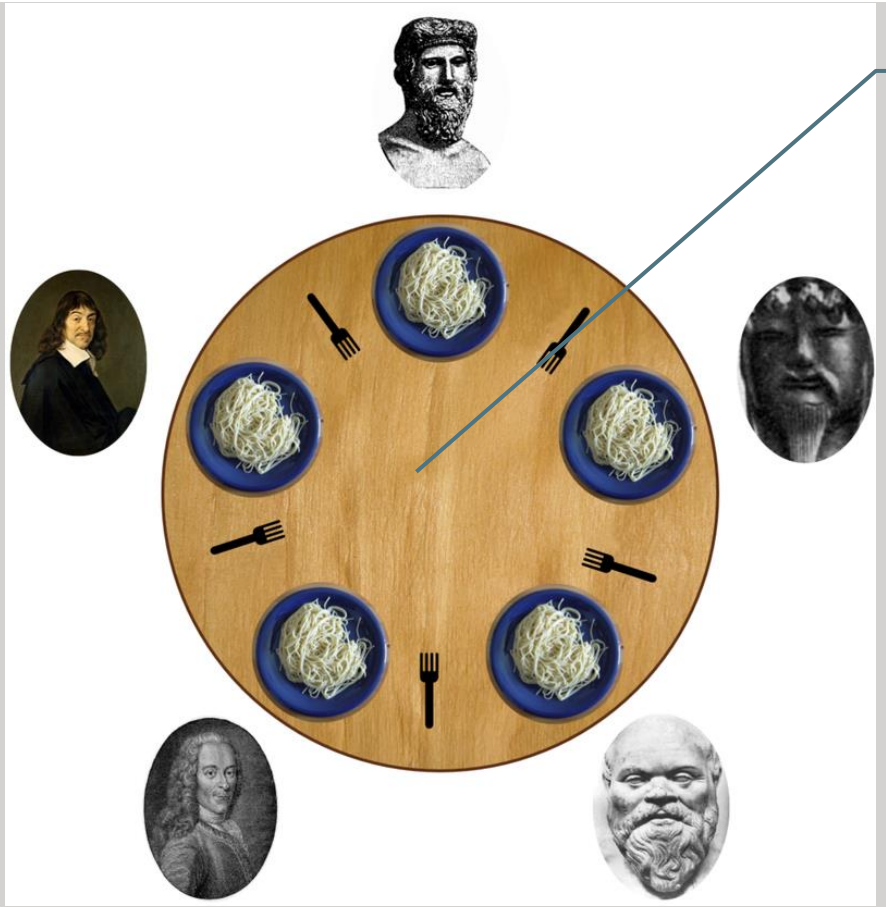
Säie / Prosessi 1	Säie / Prosessi 2

leave note A
if (noNote B){
  if (noMilk)
  buy milk
}
remove note A
	leave note B
	if (noNote A){
	  if (noMilk)
	  buy milk
	}
	remove note B

MAITOKRIISI

- Entä mikä tässä on vikana?
  - Jos konteksti (prosessin / säikeen suoritus) vaihtuu juuri väärään aikaan
  - Molemmat voi jättää lapun oveen…
    - Ja kumpikin luulee, että toinen on ostamassa maitoa!
  - Tätä muuten kutsutaan nälkiintymiseksi (starvation)
    - Yksi prosessi varaa resurssin, jota toinen tarvitsee suorittaakseen tehtävänsä => molemmat jäävät odottamaan eikä kumpikaan saa suoritusta loppuun
    - Kaikki on varmaan kuulleet ”nälkäiset filosofit”-analogian?

NÄLKÄISET FILOSOFIT



Kaikki haluaa kaksi haarukkaa; koittavat ottaa vasemmanpuoleisen. Kaikki kuolevat nälkään odottaessaan toista haarukkaa.

Perusongelma silloin, kun tehdään omia monisäikeisiä ohjelmia; Näitä ehkäistään atomisilla operaatioilla, lukoilla ja semaforeilla. Vuorontajaa ei nimittäin nämä ongelmat kiinnosta.

MITÄ TÄSTÄ LUENNOSTA PITÄÄ MUISTAA?

- Prosessien ohjaus
- Vuorontaja
- Rinnakaisten prosessien perusongelmat

HARJOITUKSET

- Uudet harjoitustehtävät on jo verkossa, harjoitusryhmät kahdesti viikossa.

LUT
University