LUENTO 7

Käyttöjärjestelmät II

VUOROTTAMINEN
YKSI CPU

Stallings, Ch 9 [Stal 05]

Mitä KJ-I:ssä / KJ-II:ssa?

KJ-I

- **Ready-jono, valitse ensimmäinen**
- **Aikaviipaletekniikka (round-robin)**
- **Prioriteetitkin mainittiin**

Seuraavaksi KJ-II:ssa

- **Vuorottamisen tasot**
- **CPU:n vuorottamisalgoritmeja**
- **Vuorottaminen UNIXissa**
- **Yhden CPU:n ympäristössä** (Ch 9) luento 7
- **SMP:ssä ja reaaliaikajärjestelmissä** (Ch 10) luento 8

Vuorottamisympäristöt, työkuorma

- **Eräajo**
  - ajetaan vaikkapa yöllä
  - työn koko osataan arvioida
    - esim. ajetaan joka yö, viikottain, kuukausittain
  - tapahtumaohjattu vuorottaminen OK
    - kunhan kaikki saadaan tehtyä
- **Interaktiivinen**
  - käyttäjä odottaa vastausta, nopea vastaus hyvä
  - vuorottajalla ei harmainta aavistusta työn kestosta
  - aikaviipaletekniikka
- **Reaaliaika**
  - aikarajat
  - ohjelmoijakin miettii suorituskykyä ja milloin KJ saa suoritusvuoron

Avoin ja suljettu työkuorma

- **Deterministinen (suljettu) työkuorma**
  - kaikki prosessit tunnetaan
  - heräämistaajuudet tunnetaan
- **Avoin työkuorma**
  - prosessien joukko vaihtelee ulkoisten tapahtumien perusteella
- **Heterogeeninen työkuorma**
  - deterministinen + avoin
    - esim. lennon valvonta, lentopintojen ohjaus
    - ”arriving missiles” –poikkeus?
      - vai deterministinen joka 100 ms?
  - eräajo + interaktiivinen + reaaliaika
  - miten sovittaa yhteen
    - jääkö avoimille reaaliaikatöille ”tarpeeksi” aikaa?
    - jääkö KJ:lle tarpeeksi aikaa?

Tavoitteita: laatu

Tbl 9.2 [Stal05]

- **Samanarvoisille prosesseille sama palvelu**
- **Priorisointia saa harrastaa**
  - turvakontrolli vs. palkanlaskenta
  - KJ prosessit vs. käyttäjän sovellukset
  - reaaliaika prosessit vs. muut
- **Interaktiiviset vs. eräajojärjestelmät**
- **Vastausaika (response time)**
  - työ annettu, milloin saadaan vastaus?
- **Läpimenoaika (turnaround time)**
  - työtä per aikayksikkö
- **Ennustettavuus (predictability)**
  - *“ei sen näin pitkään pitäisi kestää“*
  - roskien keruu, muu kj-hallinto

Linux 2.6
O(1) vuoronanto

Tavoitteita: suorituskyky

Tbl 9.2 [Stal05]

- **Ota järjestelmästä mahdollisimman paljon irti**
  - pidä CPU ja erityisesti I/O-laitteet tuottavassa työssä
    - korkea käyttöaste (CPU utilization)
  - tärkeää moniajojärjestelmissä
- **Tehokas ja reilu CPU:n käyttö**
  - läpimenoaste, läpimenovuo (throughput, work flow)
  - läpimenoaika (turnaround time, response time)
- **Reaaliaikajärjestelmä pysyy aikataulussa**
  - deadlinen ylitys voi olla harmillista tai vaarallista
    - kuvassa on häiriö, ääni/kuva epäsynkroonisia
    - potilas kuolee, lentokone tippuu, ...

Käyttöjärjestelmät II

MILLOIN

VUOROTETAAN?

Milloin?

Fig 9.2 [Stal05] Fig 9.3 [Stal05]

- **Long-term** systeemiin?
  - otetaanko uusi prosessi suoritettavaksi?
  - mahtuuko muistiin? riittääkö swap-tila?
- **Medium-term** muistiin?
  - milloin (heittovaihdettu) prosessi muistiin?
  - vapaata muistia?
  - moniajoaste?
- **Short-term** suorittimelle?
  - mille prosessille annetaan CPU?
- **I/O** I/O-laitteelle?
  - minkä prosessin I/O pyyntö palvellaan ensin?

Long-term Scheduling

Fig 9.2 [Stal05]

- **Otetaanko uusi työ suoritettavaksi?**
  - milloin työstä tulee prosessi?
  - saako koneeseen luoda uuden istunnon?
- **Ratkaisevaa: moniajoaste** medium term?
  - paljon prosesseja à kukin saa harvoin CPU:n
  - jos vähän muistia, niin onko parempi odottaa “long term” vai “medium term”? Fig 9.3 [Stal05]
  - pyritään takaamaan riittävän tasokas palvelu
  - sopiva suhde: CPU- ja I/O-sidonnaiset työt?
- **Milloin?**
  - joku prosessi päättynyt
  - CPU:n käyttöaste pudonnut
- **Mikä?**
  - First-Come-First-Serviced (FCFS)
  - joskus prioriteetteja: esim. työn koko, I/O-sidonnaisuus

Medium-Term Scheduling

- **Liittyy heittovaihtoon** Fig 9.2 [Stal05]
  - sisäänheiton ajoitus
  - prosessi tilassa Suspend&Ready tai Suspend&Wait
- **Milloin muistiin?**
  - CPU:n käyttöaste laskenut
  - vapaata muistitilaa runsaasti
- **Mikä muistiin?**
  - koko (eli odottavan prosessin muistitarve)
  - ulosheittoaika (eli odotusaika levyllä)
  - prioriteetti
- **Mikä muistista pois?**
  - ei sellainen, jolla tärkeä resurssi hallussa
    - kriittinen vaihe?
  - ei KJ prosessi?

Short-Term Scheduling

Fig 9.2 [Stal05]

- **CPU:n vuorottaminen** (scheduling, dispatching)
  - yleisterminä vuorottaminen tarkoittaa juuri tätä
- **Selvästi yleisempi kuin edelliset**
- **Milloin?**
  - keskeytyksen yhteydessä
    - jokainen keskeytys ei aiheuta vuorottamista
- **Kun nykyprosessin kyky käyttää suoritinta mennyt**
  - joutui Blocked-tilaan: I/O, synkronointi, poissulkeminen
  - poikkeustilanne
  - prosessi käyttänyt oman aikaviipaleensa
- **Suuremman prioriteetin työ valmis etenemään**
- **Kenelle vuoro seuraavaksi?**

Käyttöjärjestelmät II

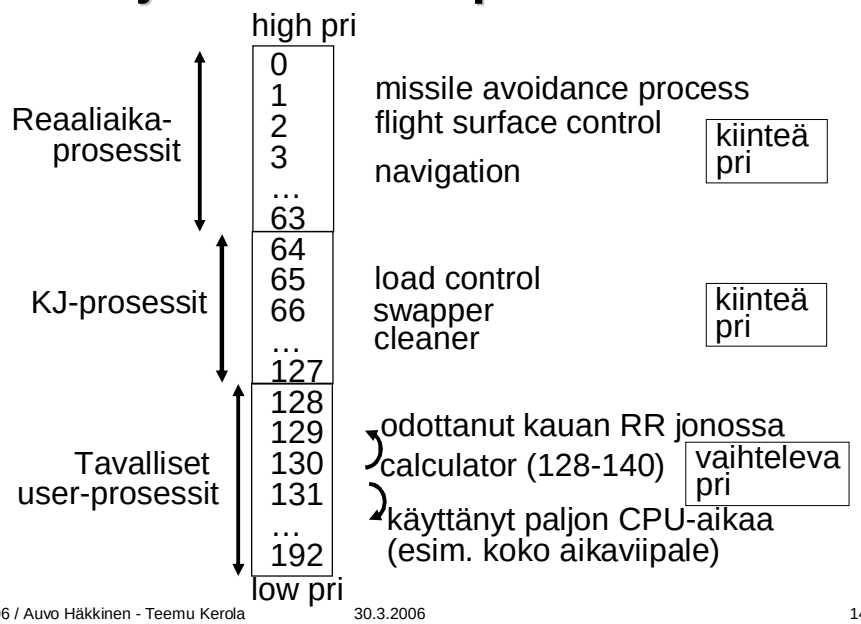
PRIORITEETTI

Prioriteetti

- **Suuremman prioriteetin prosessit ensin**
  - prioriteetti PCB:ssä / TCB:ssä
- **Kullakin prioriteetilla <u>oma Ready-jono</u>**
  - lisää aina loppuun
  - haku voi kestää, kun monta jonoa
- **Vs. <u>yksi yhteinen Ready-jonossa</u>**
  - prioriteetti määrää paikan
  - nopea haku (vain yksi jono)
  - lisäys prioriteetin mukaiseen paikkaan
    - voi olla turhan monimutkaista eli hidasta
- **Nälkiintymisvaara**
  - <u>vaihteleva prioriteetti</u> torjuu nälkiintymisen?
    - prosessin ikä
    - suoritushistoria
    - vaihtelun rajat?

Fig 9.4 [Stal05]

Kiinteä ja vaihteleva prioriteetti



Milloin vuorotus aktivoituu?

- **Nonpreemptive**
  - tapahtumaohjattu vuorottaminen
  - prosessi suorituksessa, kunnes se päättyy tai joutuu palvelupyyntönsä vuoksi blocked-tilaan
  - suoritusaikana voi silti olla keskeytyksiä ja KJ työtä!
    - scheduler ei aktivoidu
      - paitsi ehkä KJ-prosesseille?
- **Preemptive**
  - keskeyttävä vuorottaminen
    - prosessi ei voi nälkiinnyttää muita
  - suoritus keskeytetään ja prosessi Ready-tilaan, vaikka voisikin käyttää suoritinta
    - aikaviipaletekniikka
    - suuremman prioriteetin prosessi tuli Ready-jonoon
- **pre-empt: mennä edelle, ottaa itselleen etuoikeuden nojalla**

Käyttöjärjestelmät II

CPU:N VUOROTTAMIS-ALGORITMEJA

Algoritmit

- First-Come-First-Served FCFS
- Round Robin RR
- Virtual Round Robin VRR
- Shortest Process Next SPN
- Shortest Remaining Time SRT
- Highest Response Ratio Next HRRN
- Multilevel Feedback feedback

- Fair Share Scheduling FSS

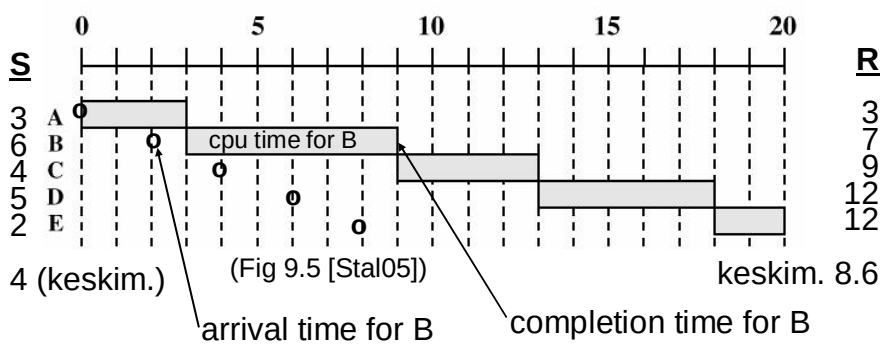
Esimerkkiprosessit

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

(Tbl 9.4 [Stal05])

- Service Time = CPU:ssa kulutettu aika
- Esimerkeissä ei mietitä I/O:n vaikutusta

FCFS – First Come First Served



- **Eräajo, tapahtumaohjattu, ei prioriteetteja**
- **Uusi prosessi Ready-jonon hännille**
- **Kun prosessi luopuu CPU:sta, vuorota seuraava**
- **Ketä suosii? Ketä ei?**

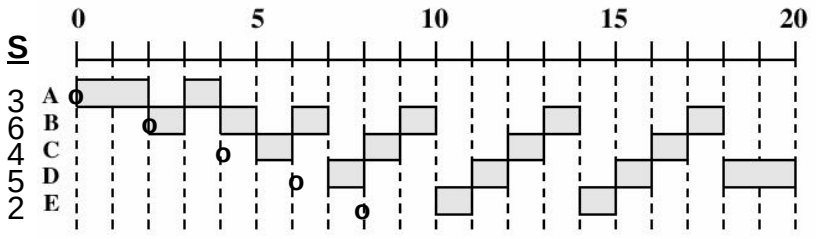
Tbl 9.4 [Stal05]

FCFS – First Come First Served

- **Läpimenoaika**
  - riippuu suoritusjärjestyksestä, muiden koosta sekä CPU-sidonnaisuudesta
- **Pienikin prosessi voi joutua odottamaan**
  - läpimenoajasta valtava osa odotusta
- **Suosii CPU-sidonnaisia**
  - muille voi tulla pitkä odotusaika
  - I/O-laitteet ehkä turhaan jouten
    - I/O kuitenkin pullonkaula
- **Järkevää ottaa mukaan prioriteetit**
  - prioriteetin perusta?
    - prosessin koko (suoritin aika)?
    - I/O-sidonnaisuus?

Fig 9.5 [Stal05]

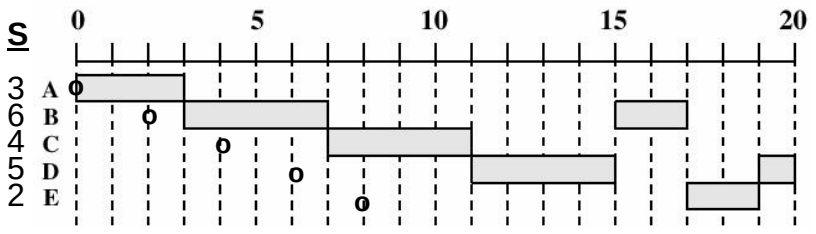
RR – Round Robin



4 (keskim.) (Fig 9.5 [Stal05]) keskim. 10.8

- **Aikaviipaletekniikka (q=1), keskeytyvä (preemptive)**
- **Kukin Ready-prosessi saa vuorollaan aikaviipaleen**
- **Vuorottaminen, kun viipale käytetty <u>tai</u> kun prosessi joutuu Blocked-tilaan**

RR – aikaviipaleen vaikutus



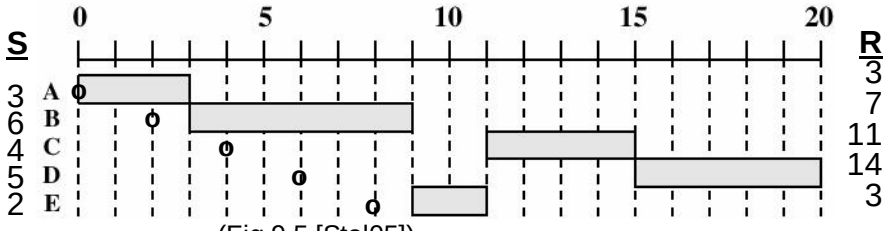
4 (keskim.) (Fig 9.5 [Stal05]) keskim. 10.0

- **Aikaviipaletekniikka q = 4**
  - prosessin vaihtoaikaa ei huomioitu!
- **Suosiiko pitkä aikaviipale cpu- vai I/O-sidonnaisia?**
- **Suosiiko pitkä aikaviipale lyhyitä vai pitkiä töitä?**

RR – Round Robin

- **Aikaviipaleen pituus** — Fig 9.5 [Stal05]
  - lyhyt: prosessin vaihdot vievät CPU-aikaa
  - pitkä: interaktiivisen työn vastausaika lyhyt, jos yksi aikaviipale riittää
- **Suosii hieman CPU-sidonnaisia**
  - I/O-sidonnainen ei ehkä käytä koko viipaletta
  - I/O-sidonnainen saa suhteessa harvemmin CPU:n
    - miksi?
- **Virtual RR**
  - ready-jonon apujono (Auxiliary Ready Queue), jonne I/O-odotuksesta
    - prioriteetti I/O sidonnaisilla — Fig 9.7 [Stal05]
  - aja ensin apujonossa olevat prosessit
  - aikaviipale vain edellisellä kerralla käyttämättä jäänyt osa, sitten normaaliin Ready-jonoon

SPN – Shortest Process Next



(Fig 9.5 [Stal05])

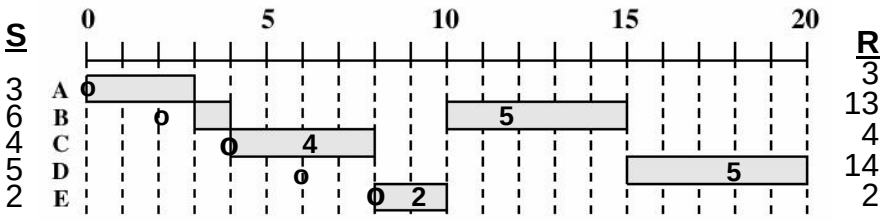
4 (keskim.) keskim. 7.6

- **Tapahtumaohjattu (siis non-preemptive)**
- **Vuorota se, joka käyttää lyhimmän ajan CPU:ta kerrallaan**
  - I/O-sidonnaiset ensin
  - mistä tietää?

SPN – Shortest Process Next

- **Nälkiintymisvaara**
  - iso jää aina pienten jalkoihin
- **Isojen läpimenoaika vaikea ennustaa**
- **Erätyö: käynnistäjä arvioi työn kestoajan**
  - jos työ laitettiin väärään eräajoluokkaan, KJ saattaa katkaista työn
    - miksi? (kulutti liikaa aikaa estimaattiin nähden)
    - käynnistettävä uudelleen isompien luokassa
- **Interaktiivinen: KJ laskee keskim. CPU:n käyttöaikaa**
  - painottaa viimeksi havaittuja aikoja (T_{n-1})
  - estimoi $S_n^{estim} = \alpha T_{n-1} + (1 - \alpha) S_{n-1}^{estim}$ esim $\alpha = 0.8$
- **Ei sovellu osituskäyttöympäristöön**

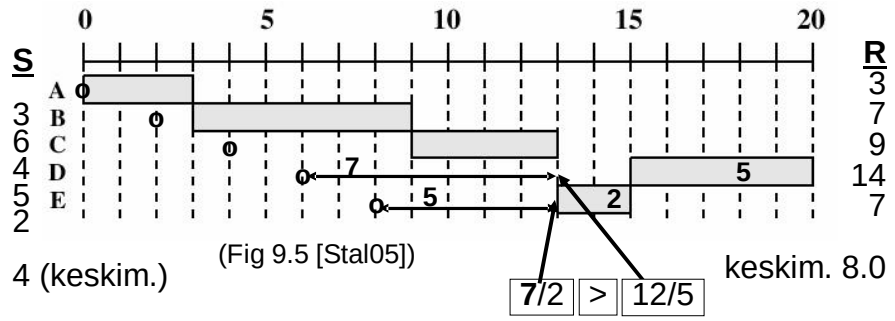
SRT – Shortest Remaining Time



4 (keskim.) (Fig 9.5 [Stal05]) keskim. 7.2

- **Aikaviipaleversio edellisestä (keskeytyvä)**
  - tilanne arvioidaan uudelleen joka aikaviipaleelle
- **Arvioitava prosessin jäljellä oleva ajantarve**
- **Ei sovi interaktiiviseen ympäristöön**

HRRN – Highest Response Ratio Next



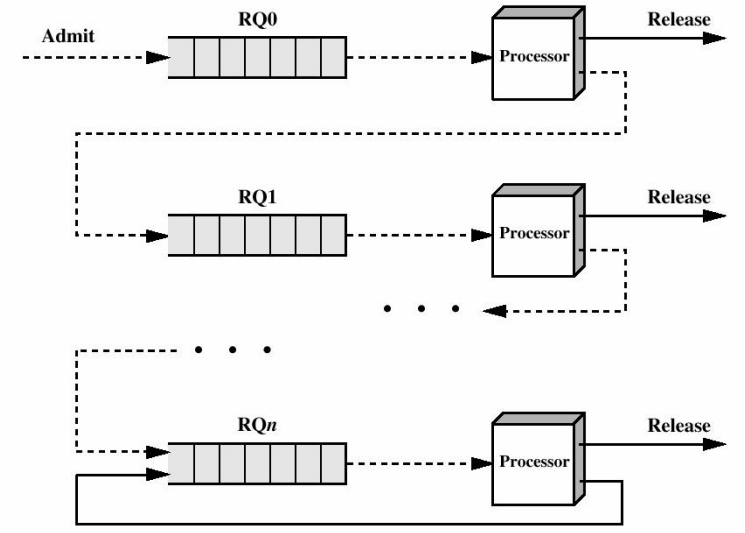
- **Tapahtumaohjattu (siis non-preemptive)**
- **Minimoi läpimenoaikaa (huomioi historia)**
- **Vuorota se, jolla huonoin suhteellinen vasteaika, ts. se, jolla suurin suhdeluku:**

$$\text{response ratio} = \frac{\text{time spent waiting CPU + expected service time}}{\text{expected service time}}$$

HRRN – Highest Response Ratio Next

- **Suosii hieman lyhyitä töitä**
  - ei silti nälkiintymisvaaraa
- **Dynaaminen prioriteetti**
  - ready-jonossa odottelu kasvattaa prioriteettia
- **Jäljellä olevaa aikaa ei voi tietää**
  - arviot menneisyyden perusteella
  - käyttäjän antama arvio työn koosta
    - ei sovi interaktiiviseen ympäristöön

Multilevel Feedback

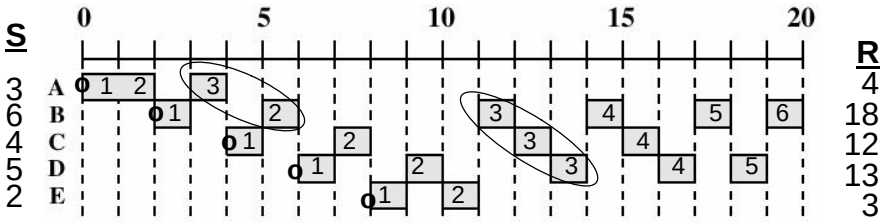


(Fig 9.10 [Stal05])

Feedback

- **Dynaaminen prioriteetti**
  - ‘Rankaisee’ pitkään pörränneitä prosesseja, aikaviipaloitu
- **Useita Ready-jonoja**
  - RQ0: pura aikaviipaleittain, FCFS, siirrä seuraavaan jonoon
  - RQ2..RQn-1: pura aikaviipaleittain, FCFS, siirrä seuraavaan jonoon
  - RQn: pura aikaviipaleittain, RR, pidä samassa jonossa
- **Prosessi kulkeutuu lopulta RQn-jonoon, josta se aikanaan valmistuu**
- **Nälkiintymisvaara**
  - vuorottaa alemmassa jonossa olevat aina ensin
- **Useita variaatioita**
  - esim. alemmissa jonoissa pitempi aikaviipale (esim. 1, 2, 4, …)
  - palaa blocked-tilasta samaan jonoon

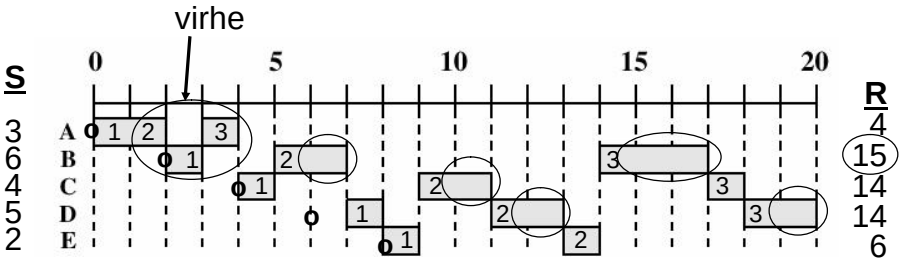
Feedback q=1



4 (keskim.) (Fig 9.5 [Stal05]) keskim. 10.0

- **Jonot RQ0, RQ1, RQ2, …**
- **Aikaviipale q=1**
- **Ei vaadi etukäteisarvioita CPU-ajan tarpeesta**
- **Pitkät työt voivat kestää kauan**

Feedback q=2^i



4 (keskim.) (Fig 9.5 [Stal05]) keskim. 10.6

- **Jonot RQ0, RQ1, RQ2, …, RQi**
- **Aikaviipale q=2^i**
- **Vaikutus, vrt. q=1**

Fig 9.5 [Stal05]

Yhteenveto

Tbl 9.3 [Stal05]

	Selection Function	Decision Mode	Throughput	Response Time	Overhead	Effect on Processes	Starvation
FCFS	max[w]	Nonpreemptive	Not emphasized	May be high, especially if there is a large variance in process execution times	Minimum	Penalizes short processes; penalizes I/O bound processes	No
Round Robin	constant	Preemptive (at time quantum)	May be low if quantum is too small	Provides good response time for short processes	Minimum	Fair treatment	No
SPN	min[s]	Nonpreemptive	High	Provides good response time for short processes	Can be high	Penalizes long processes	Possible
SRT	min[$s - e$]	Preemptive (at arrival)	High	Provides good response time	Can be high	Penalizes long processes	Possible
HRRN	$\max\left(\frac{w+s}{s}\right)$	Nonpreemptive	High	Provides good response time	Can be high	Good balance	No
Feedback	(see text)	Preemptive (at time quantum)	Not emphasized	Not emphasized	Can be high	May favor I/O bound processes	Possible

w = time spent ~~in system so far~~ waiting ~~and executing~~
e = time spent in execution so far
s = total service time required by the process, including e

Käyttöjärjestelmät II

UNIX

SVR3 / BSD4.3

VUOROTTAMINEN

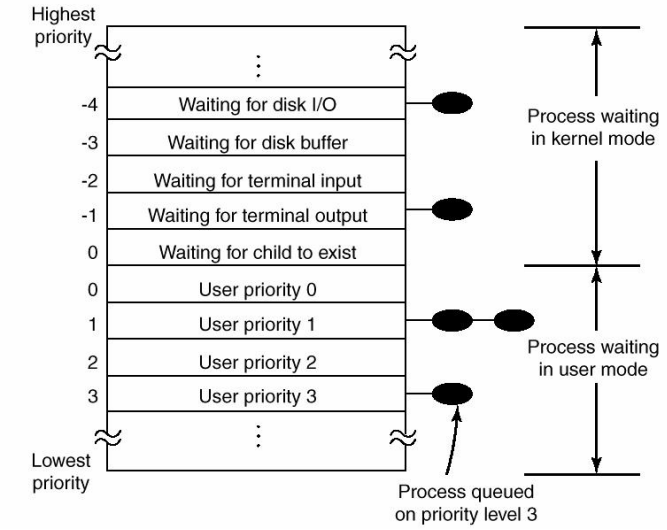
UNIX: Vuorottaminen

- **Interaktiivinen ympäristö**
  - ei varsinaista eräajoa, ei erätyöjonoja
  - at-komento ohjelmien ajamiseksi myöhemminkin
  - crontab periodisille töille

cron = “chronogram” deamon?
tab = table

- **Pyrkii hyvään vastausaikaan**
  - taustaprosesseilla huono prioriteetti
- **Aikaviipaleet, Round-Robin**
- **Multilevel feedback**
  - prioriteeteilla omat Ready-jonot
  - tyhjentää suurimman prioriteetin jonon ensin
  - dynaaminen prioriteetti à ei nälkiintymistä

UNIX: Ready-jonot



(Fig 10-11 [Tane01])

UNIX: Prioriteetti

- **Kiinteä perusprioriteetti sekä nice-arvo**
  - käyttäjä voi pienentää prioriteettia nice-komennolla
  - pitää prioriteetin siististi tietyllä arvoalueella
- **Laskee uuden prioriteetin sekunnin välein**
  - iso aikaviipale!
- **CPU:n käyttö vaikuttaa uuteen arvoon**
  - käyttö: prioriteetti putoaa
  - odottaa kauan: prioriteetti kasvaa
- **Suosii I/O-sidonnaisia prosesseja**
  - tavoite: I/O-laitteiden tehokas työllistäminen

UNIX: multilevel feedback

- **CPU_usage = CPU:n käyttö äskettäin**
  - laskuri PCB:ssä
- **Älä rankaise liikaa aiemmasta käytöstä**
  - puolita ennen prioriteetin laskentaa, ja sitten taas

```
CPU_usage = CPU_usage/2
Pri = Base + (CPU_usage/2)+ Nice
    (jos ei cpu_usage = 0, niin ei muutosta)
```

- **Pieni arvo = suuri prioriteetti**
- **Esimerkki**

Fig 9.17 [Stal05]

  - Base=60, Nice = 0
  - päivitä counter (CPU_usage) 60 kertaa/sek
    - kellolaite keskeytys 16.7 ms välein?
    - otanta: kenellä suoritin keskeytyshetkellä?
  - päivitä prioriteetti sekunnin välein

Fair-Share Scheduling

- **Tutki myös kuka prosessin omistaa (owner)**
  - ettei hulivili voi tukkia järjestelmää…
- **Käsittele yhden käyttäjän prosesseja / säikeitä ryhmänä**
  - ryhmän vaikutus “nice” termin asemesta
  - vuorottelu edelleen prosessi- / säietasolla
  - pidettävä myös kirjaa paljonko ryhmä saanut CPU:n kokonaisajasta (GCPU_counter)
  - ryhmällä voi olla paino W, joka määrää millaisen osuuden se saa koko (cpu-aika) kakusta
- **Käytössä useissa UNIX-järjestelmissä**
  - HP-UX, IBM AIX WLM, Sun Solaris SRM
  - ryhmä voi perustua käyttäjään tai prosessiin tai sovellukseen

FSS - Fair-Share Scheduling

- **Prioriteetin määrittäminen**

```
CPU_counter  = CPU_counter/2
GCPU_counter = GCPU_counter/2
Pri = Base + CPU_counter/2 + GCPU_counter/(4*W_group)
```

iso W_{group}
à pieni painoarvo
GCPU_counter:lla
à pieni Pri arvo
à iso prioriteetti

Esimerkissä

- **Base = 60**
- **W_A = 0.5 ja W_{B+C} = 0.5** ($W_A + W_{B+C} = 1$)
- **päivitä laskurit 60 kertaa sekunnissa**
- **päivitä prioriteetti sekunnin välein**

Fig 9.16 [Stal05]

Kertauskysymyksiä

- Milloin vuorotetaan?
- Tapahtumaohjattu (non-preemtive) vs. keskeytyvä (pre-emptive) vuorottaminen
- Miten FCSF ja Round-Robin algoritmit eroavat toisistaan?
- Millaisiin tilanteisiin ne sopivat?
- Mikä FSS algoritmin perusidea?
- Mitä hyötyä on käyttää prioriteetteja?
- Mitä hyötyä on vaihtelevan prioriteetin käytöstä?