Operacinės sistemos P175B304

05T

doc. Ingrida Lagzdinytė-Budnikė

2018-03-05

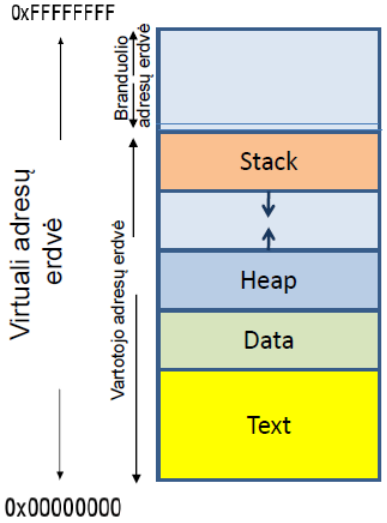
2018-03-06

Paskaitos turinys

- Procesų valdymas
  - **Procesas. Jo būvis, kontekstas. Persijungimas nuo vieno proceso prie kito.**
  - Gijos, realizacijos modeliai. Proceso-gijos skirtumai.
  - Procesų vykdymo planavimas. Tikslai, mechanizmai, naudojimo sąlygos.
  - Tarprocesinė (IPC) komunikacija, klasikinės IPC komunikacijos problemos.

Praėjusį kartą kalbėjome:

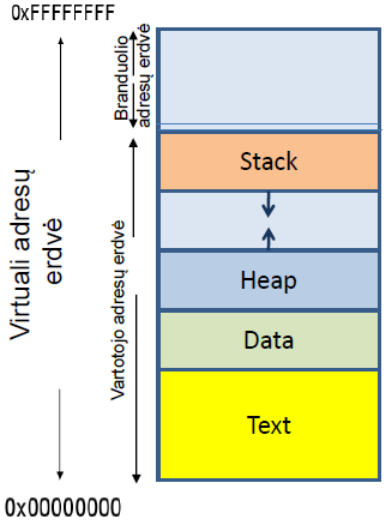
Procesas atmintyje

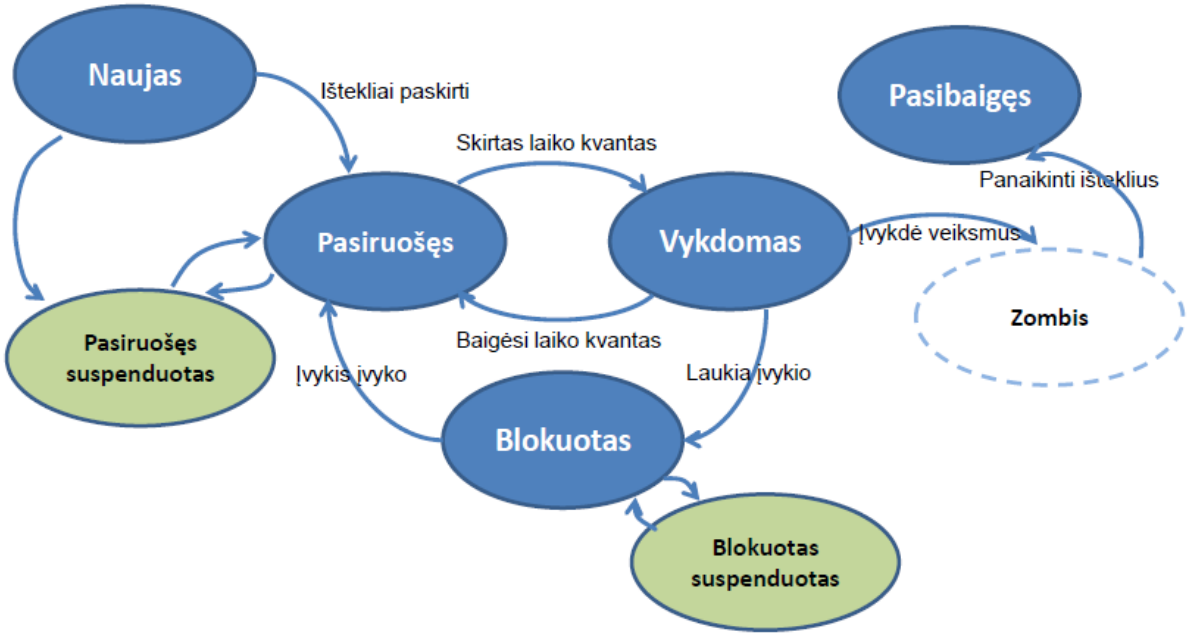


- Steko sritis (stack). Saugomi lokalių funkcijų kintamieji
- Heap sritis. Saugomi dinamiškai kuriami objektai. (C – `malloc()`, C++ - `new`).
- Duomenų sritis (data). Saugomi globalūs kintamieji
- Programos kodo sritis (text). Saugomas objektinis vykdomasis programos kodas

Praėjusį kartą kalbėjome:

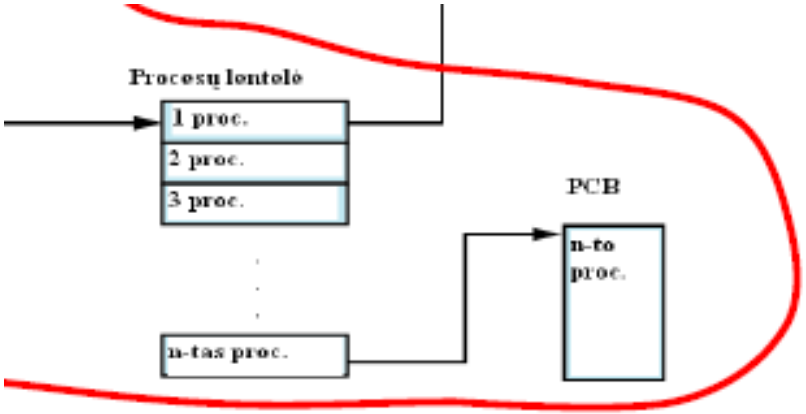
Procesas atmintyje





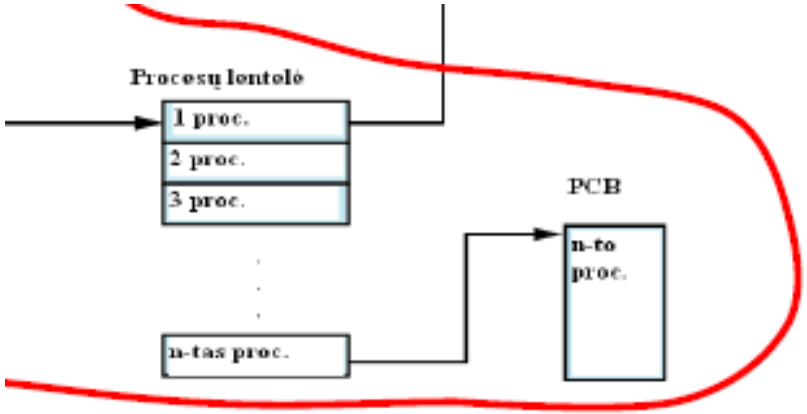
Praėjusį kartą kalbėjome:

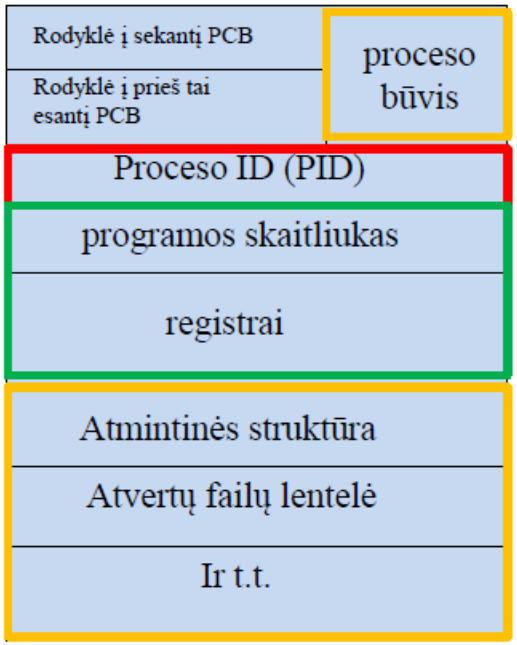
- Procesų lentelė
- Proceso kontrolės blokas
- Procesų eilės



Praėjusį kartą kalbėjome:

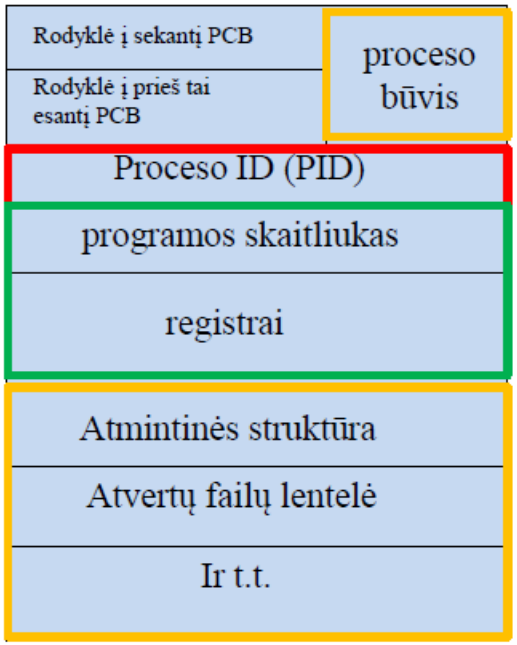
- Procesų lentelė
- Proceso kontrolės blokas
- Procesų eilės

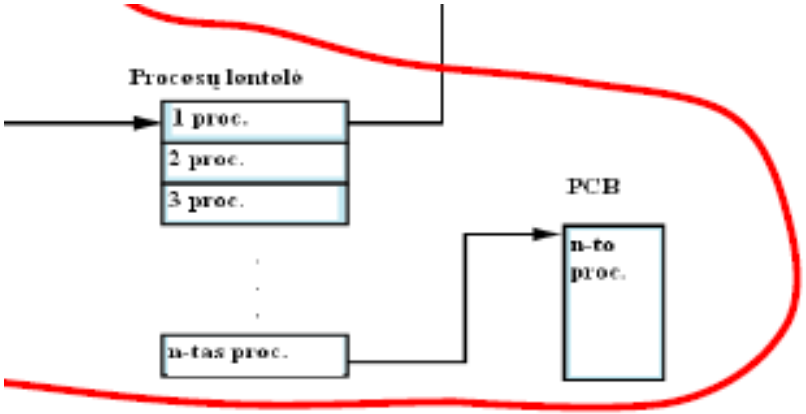


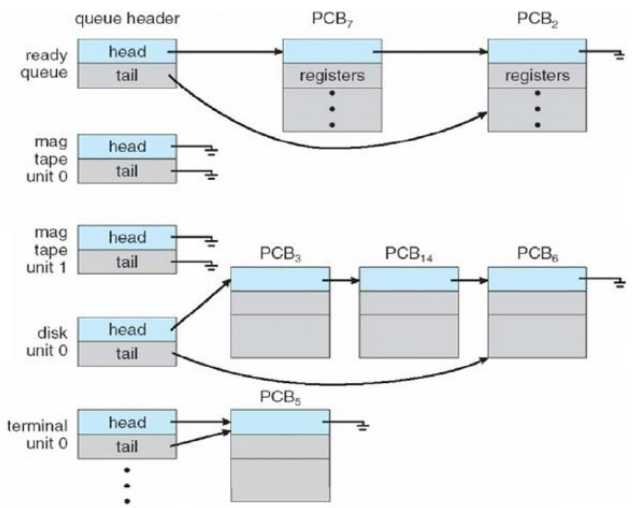


Praėjusį kartą kalbėjome:

- Procesų lentelė
- Proceso kontrolės blokas
- Procesų eilės

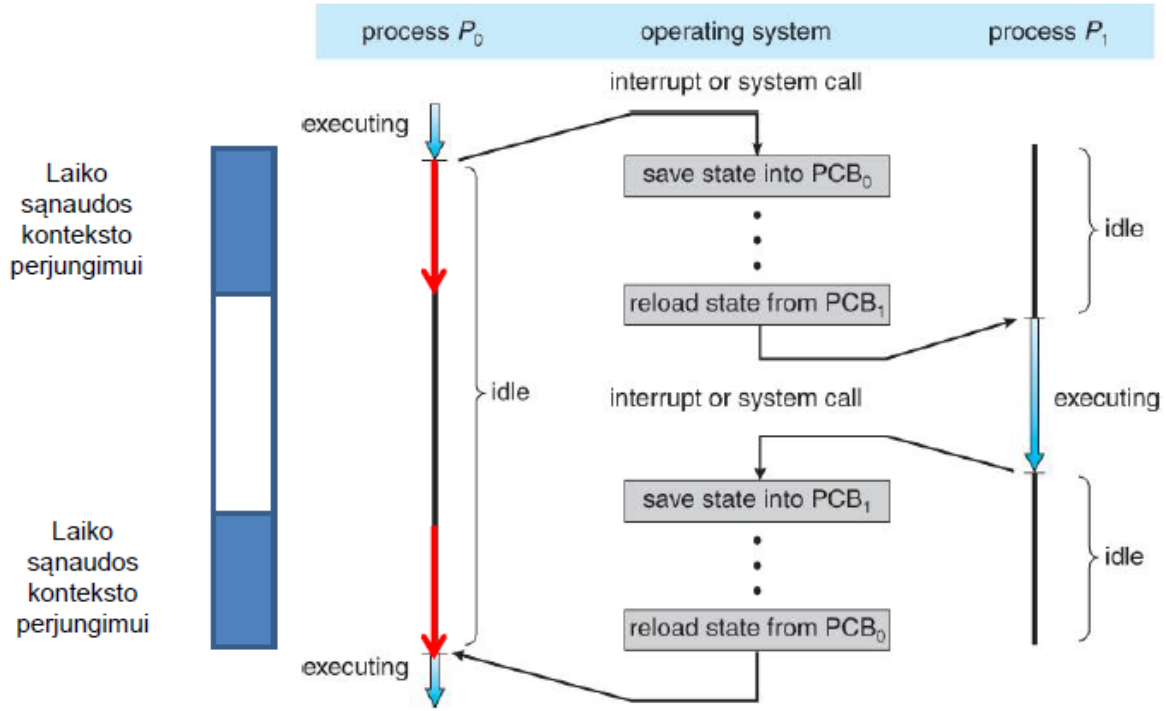






Praėjusį kartą kalbėjome:

Perėjimo nuo vieno proceso prie kito procedūra



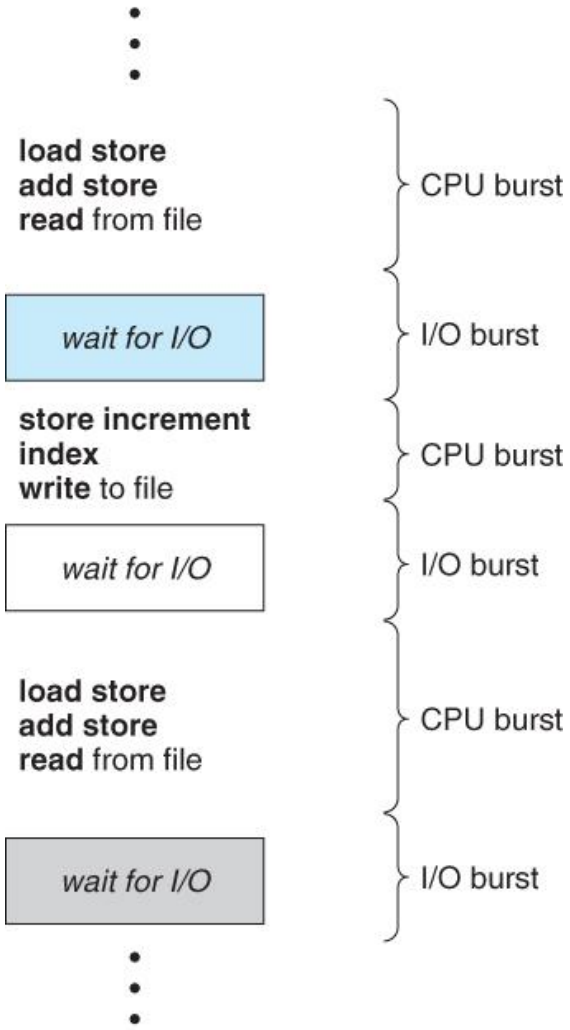
Paskaitos turinys

- Procesų valdymas
  - Procesas. Jo būvis, kontekstas. Persijungimas nuo vieno proceso prie kito.
  - Gijos, realizacijos modeliai. Proceso-gijos skirtumai.
  - **Procesų vykdymo planavimas. Tikslai, mechanizmai, naudojimo sąlygos.**
  - Tarprocesinė (IPC) komunikacija, klasikinės IPC komunikacijos problemos.

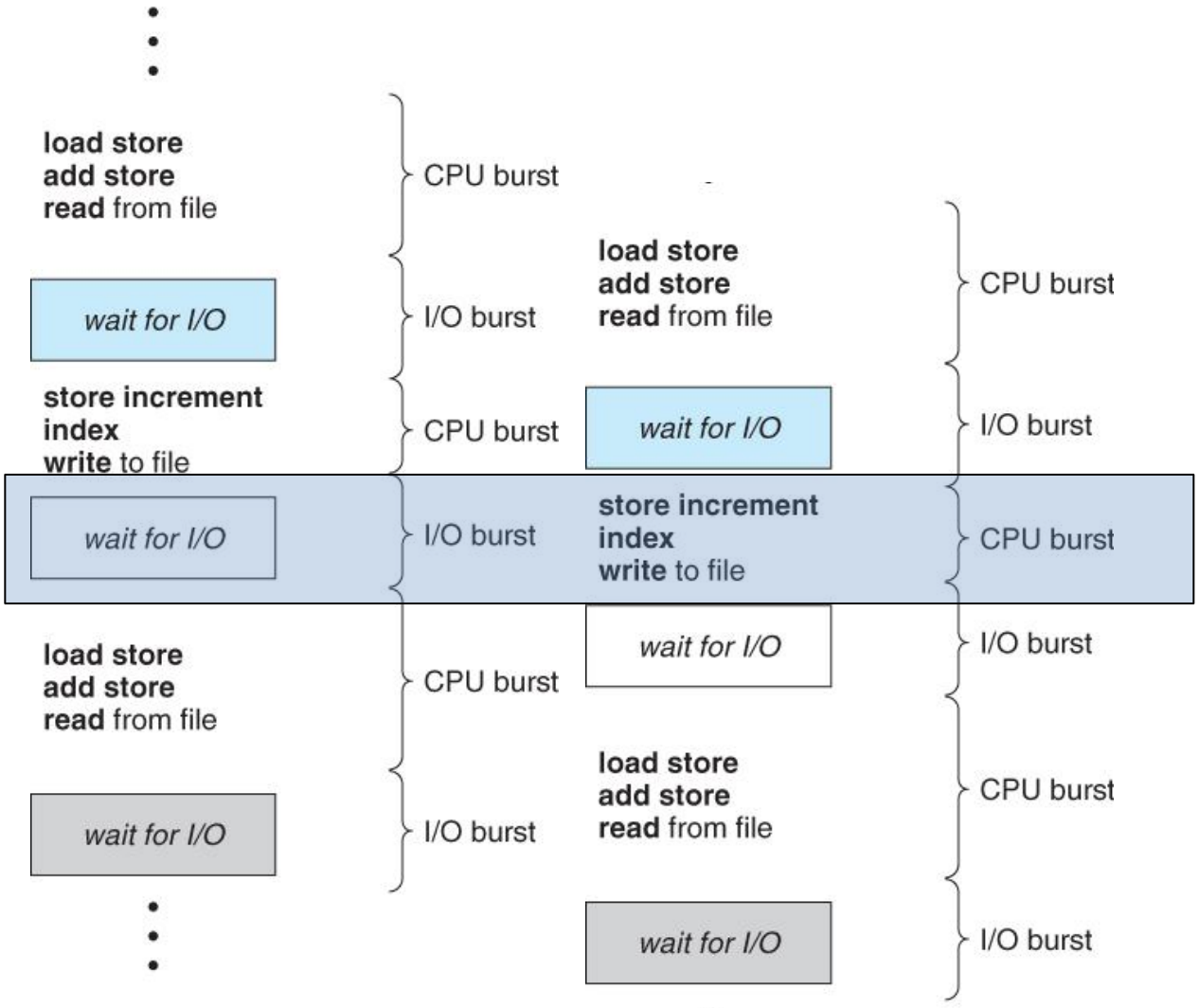
Įvadas ir prielaidos

- Pirmieji su šia sritimi susiję tyrimai ir pasiūlymai – apie 1970 m.
- Daromos išankstinės prielaidos:
  - Sistemos naudotojas vykdo 1 programą, t.y. vieną procesą
  - Procesas turi vieną giją
  - Procesai yra nepriklausomi
- Prielaidos neatspindi tikrovės, tačiau leidžia supaprastinti nagrinėjamą problemą.

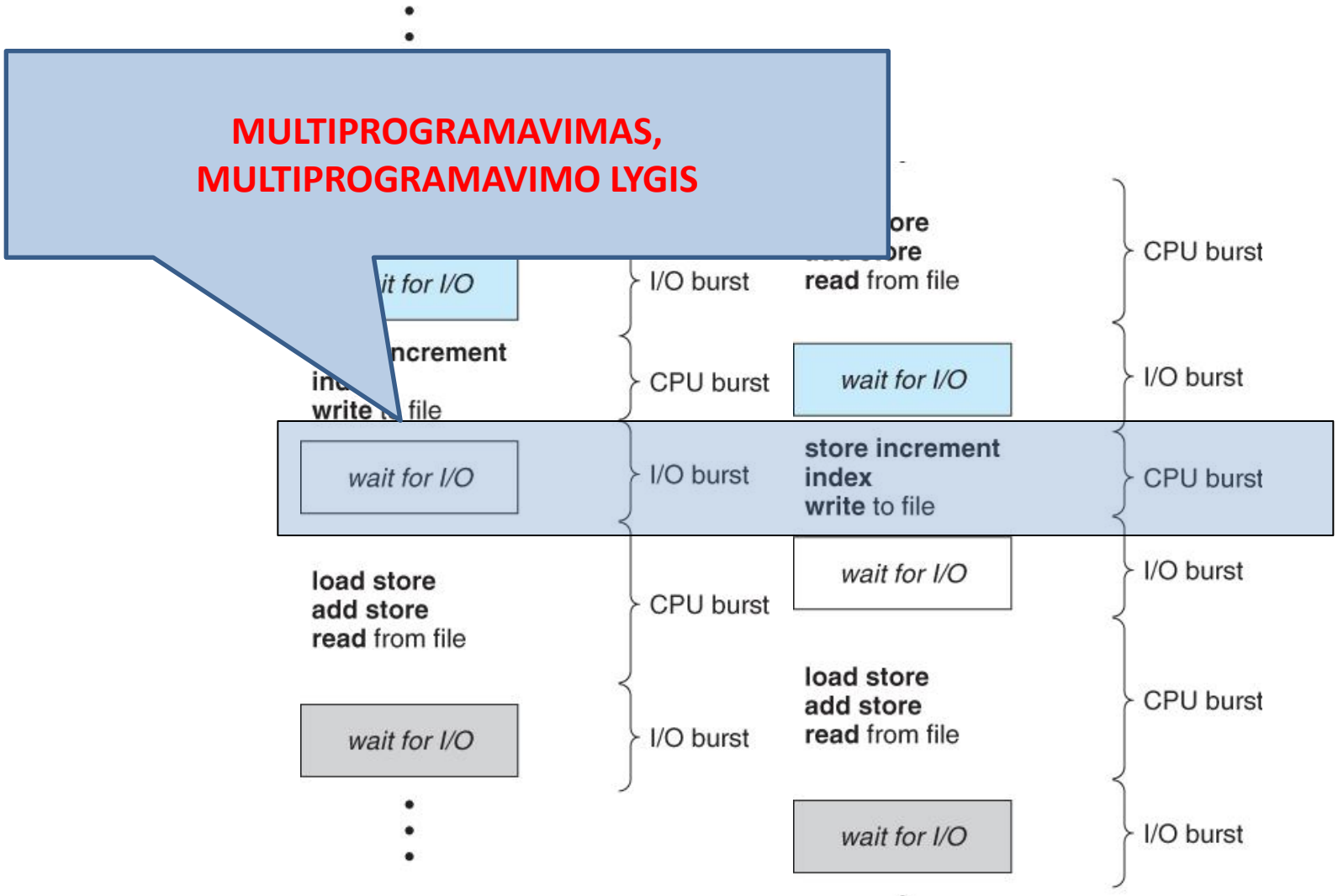
CPU-IO ciklas



CPU-IO ciklas

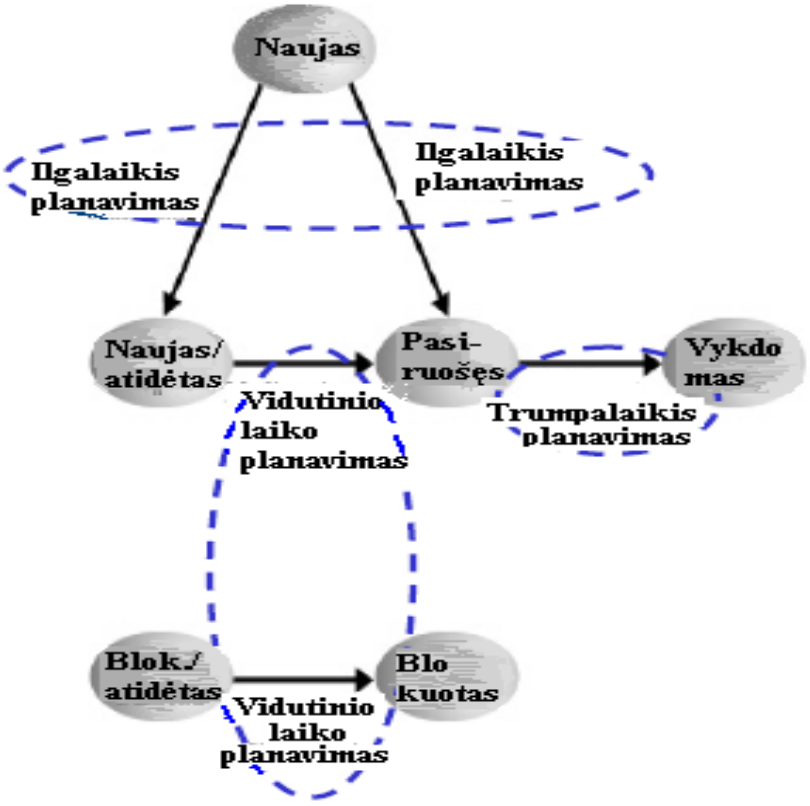


CPU-IO ciklas



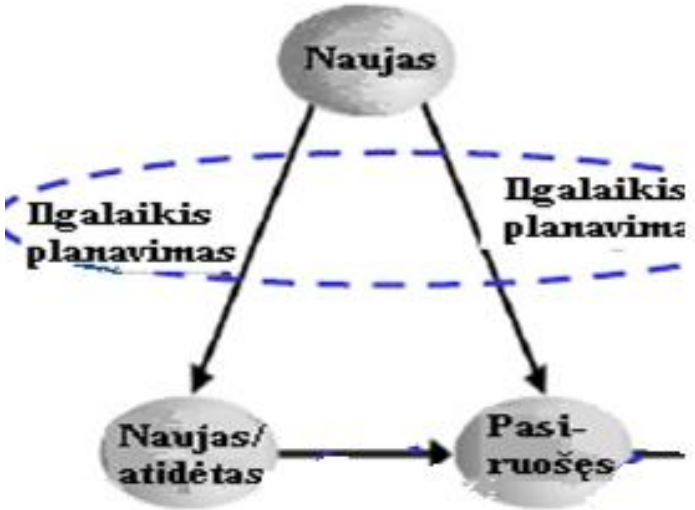
trijų tipų planavimas

- Ilgalaikis planavimas (*angl. job scheduler*).
- Vidutinio laiko planavimas (*angl. swapper*).
- Trumpalaikis planavimas (*cpu scheduler*).

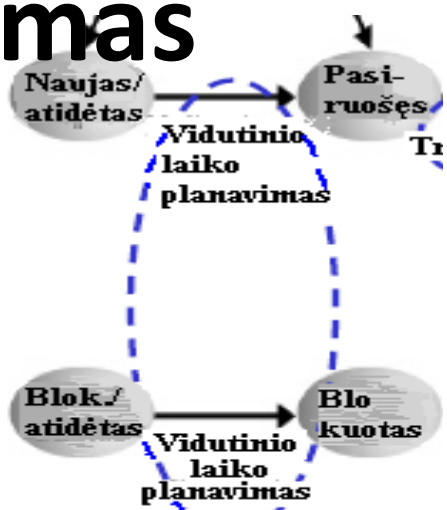


Ilgalaikis planavimas

- Tai **užduočių** planuoklė (angl. job scheduler)
- Sprendžia:
  - kada ir kokia tvarka priimti į sistemą naujus procesus
  - kuris procesas bus įtrauktas į eiles (sąrašus) „**Naujas**“ ir “**Pasiruošęs**”.
- Aktyvuojama:
  - Kažkuriam iš procesų ***pasibaigus***.
  - Kai procesorius ***prastovi*** ilgiau nei tam tikrą nustatytą laiko intervalą.
- Užklausos gali būti atmestos, jei:
  - yra ***perpildymo*** būsena
  - Didelis puslapių mainų ***aktyvumas***.



Vidutinio laiko planavimas



- "Swapper" procesas (angl. memory scheduler)
- Sprendžia:
  - kada ir kokį procesą reikia ***atidėti*** (iškelti iš pagrindinės atmintinės)
  - kada ir kokį procesą reikia ***suaktyvinti*** (įkelti į pagrindinę atmintinę).
- Aktyvuojama:
  - Procesui perėjus į blokuoto būseną
  - Nesant pakankamai vietos visiems procesams pagrindinėje atmintinėje
- Ilgalaikio ir vidutinio laiko planavimo mechanizmai kontroliuoja ***multiprogramavimo lygį***. t.y. apsprendžia vienu metu vykdomų procesų kiekį.

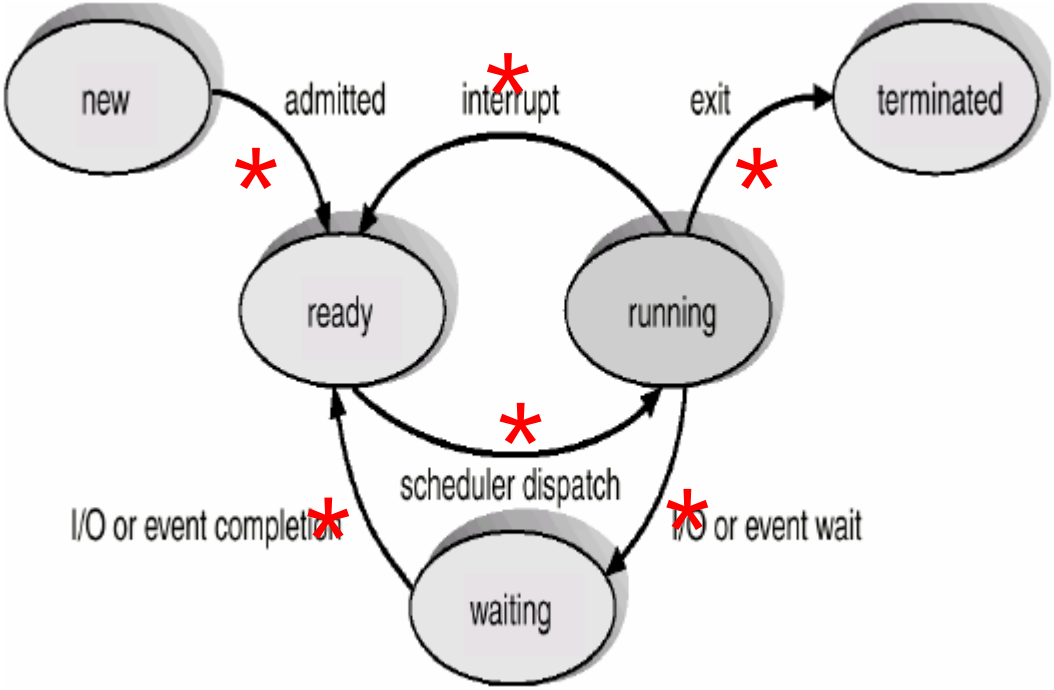
Trumpalaikis planavimas

- ***Sprendžia, kurio proceso vykdymui bus skiriamas CPU- tai CPU planuoklė***.
- Šis planavimas vykdomas įvykus vienam iš šių įvykių:
  - ***Laikrodžio*** mechanizmo generuota pertrauktis.
  - I/O ***pertrauktis***.
  - Sutiktas OS ***kvietinys***
  - ***Signalas***
- Šio planavimo rezultate iš pasiruošusių procesų eilės yra išrenkamas vienas procesas, jo būvis tampa „**Vykdomas**“, jo kodas pradedamas vykdyti.

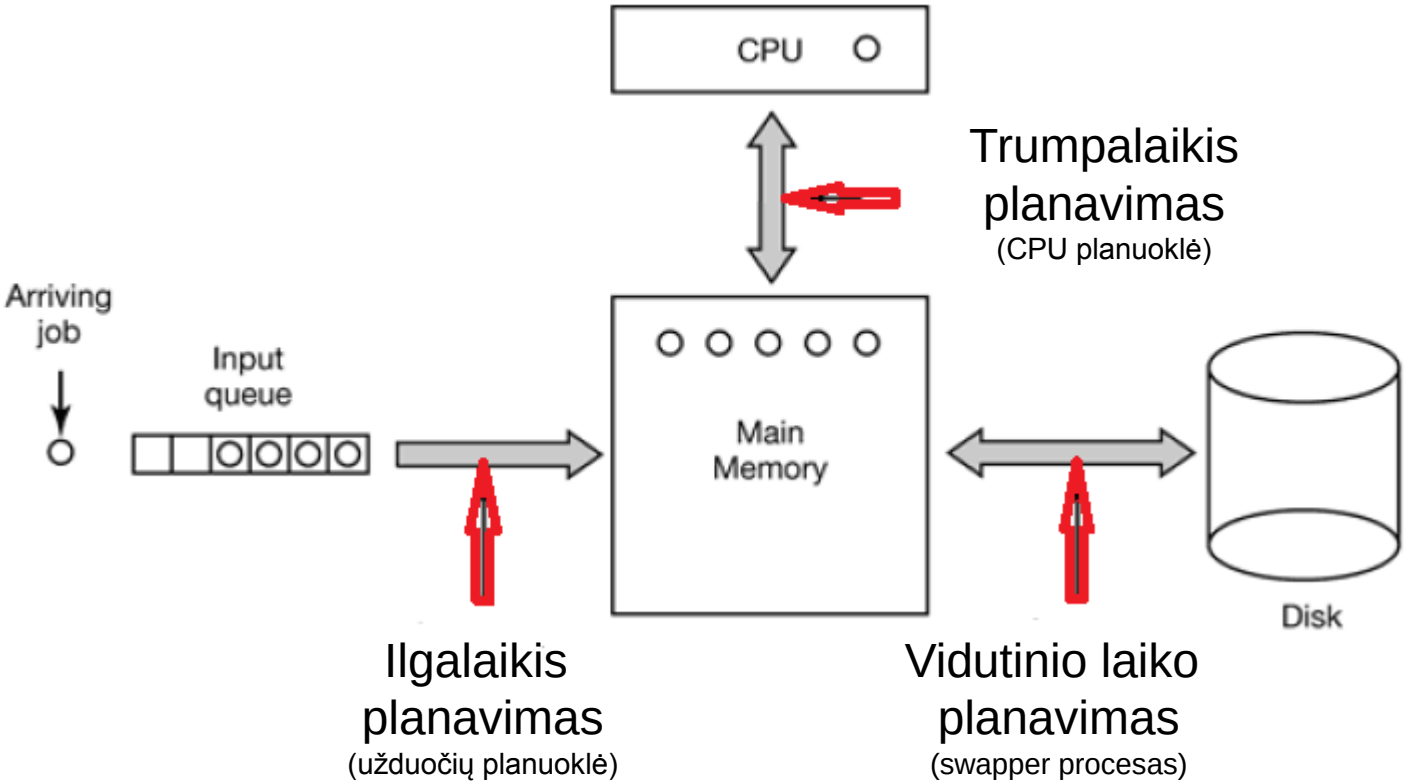
Pav. Solaris OS vertina pertrauktis, kurių lygis svyruoja nuo 0 iki 15. Tipiniai:

- prog-įrangos
- SCSI/FC diskų (3)
- Juosta (Tape), Ethernet
- Video/grafika
- Laikrodis (10)
- Nuoseklios komunikacijos
- Realaus laiko CPU laikrodis
- “Nonmaskable” pertr. (15)

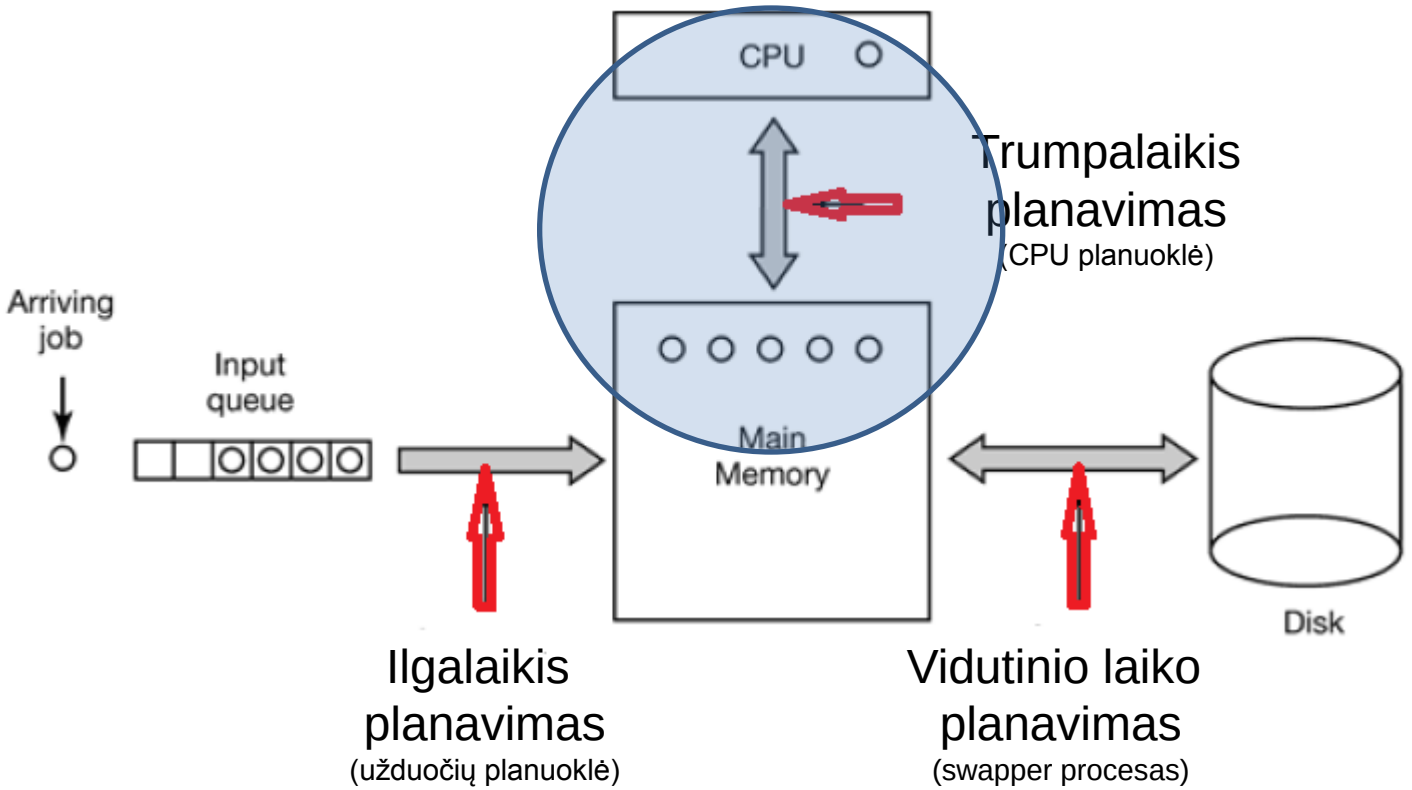
Planavimo taškai



Planuoklių išsidėstymo schema



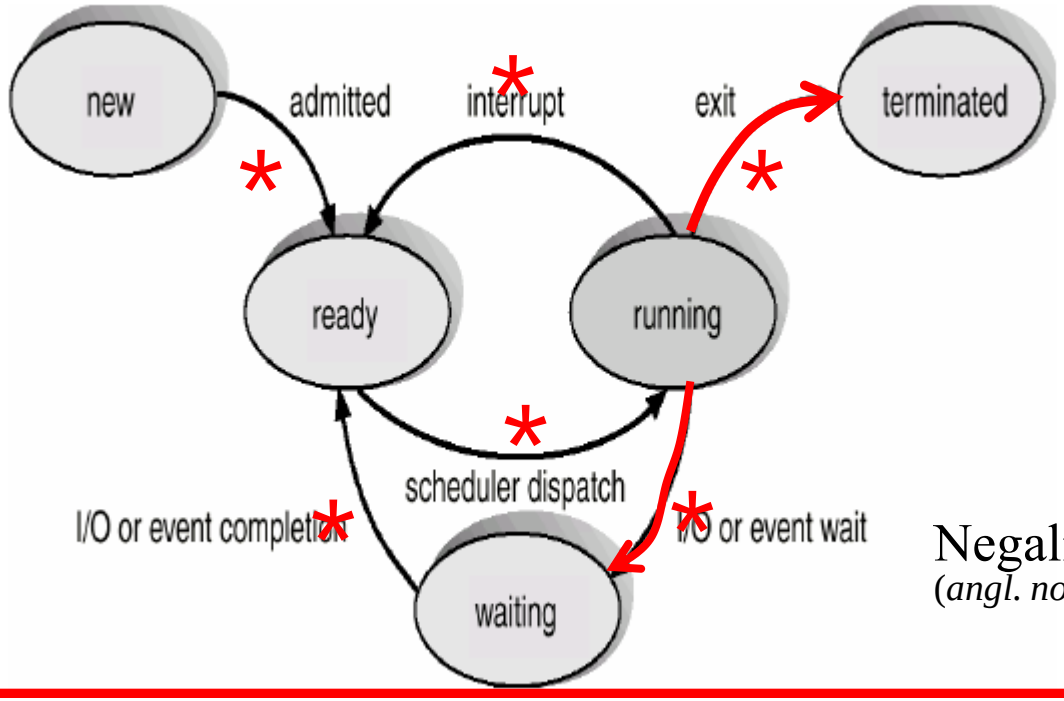
Planuoklių išsidėstymo schema



CPU planuoklė

- Modulis, kuris manipuliuoja procesų eilėmis, perkeldamas užduotis iš vienos eilės į kitą.
- Planavimo algoritmas apsprendžia, kurią užduotį išrinkti vykdyti sekančia.
- Dispečeris
  - Dispečerio modulis perduoda CPU kontrolę planuoklės išrinktam procesui, tai apima:
    - Procesų konteksto ***perjungimą*** (perėjimą nuo vieno proceso prie kito, procesų būsenų pakeitimą)
    - ***Perėjimą*** į *vartotojo* būvį (angl. user-mode)
    - ***Peršokimą*** į atitinkamą vietą vartotojo programoje
  - Dispečeriavimo vėlinimas: laikas, kurio reikia dispečeriui vieno proceso sustabdymui ir kito paleidimui

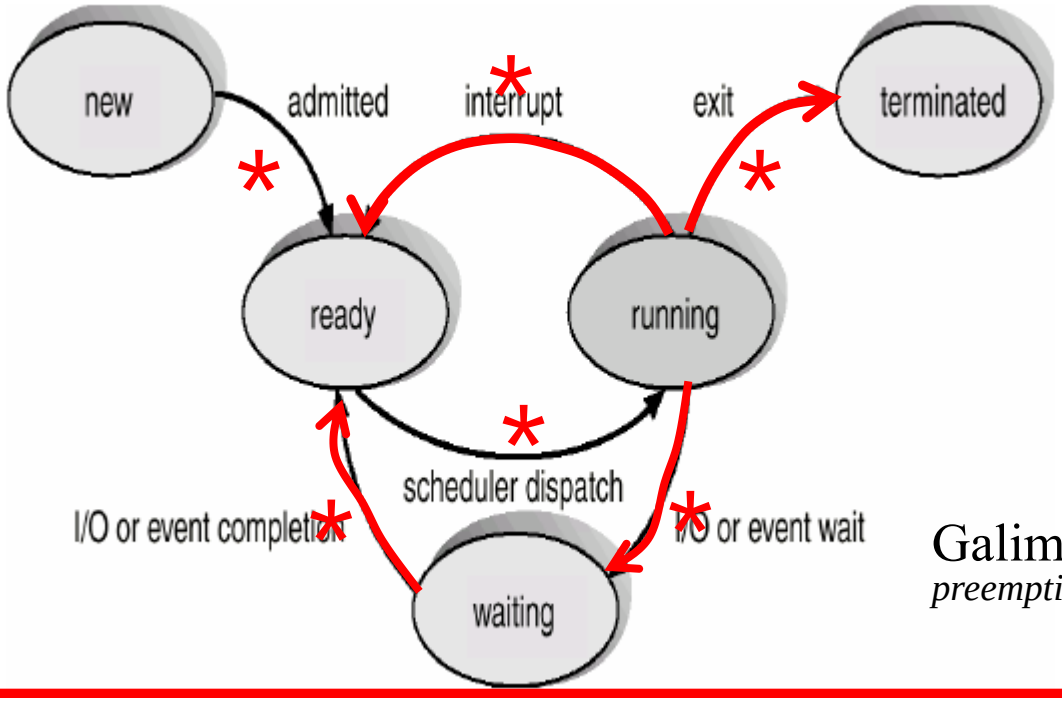
CPU Perėmimas



Negalimas perėmimas
(*angl. non-preemptive scheduler*)

1. Vykdomas procesas pereina iš “running” būsenos į “waiting” esant I/O užklausai
2. Vykdomas procesas pasibaigia
3. Laikrodžio pertrauktis iššaukia planuoklę dėl sprendimo, palikti procesorių einamajam procesui ar perkelti jį į “ready” eilę ir vykdyti sekantį procesą
4. Gavus atsakymą į I/O užklausą procesas iš “waiting” būsenos pereina į “ready” būseną

CPU Perėmimas



Galimas perėmimas (*angl. preemptive scheduler*)

1. Vykdomas procesas pereina iš “running” būsenos į “waiting” esant I/O užklausai
2. Vykdomas procesas pasibaigia
3. Laikrodžio pertrauktis iššaukia planuoklę dėl sprendimo, palikti procesorių einamajam procesui ar perkelti jį į “ready” eilę ir vykdyti sekantį procesą
4. Gavus atsakymą į I/O užklausą procesas iš “waiting” būsenos pereina į “ready” būseną

Planavimo politika - tikslai

- Minimizuoti [visą] atsakymo laiką:
  - Minimizuoti proceso apdorojimo laiką sistemoje (laikas pasiruošusių procesų eilėje, išskirti reikiamą CPU kvanto dydį)

Planavimo politika - tikslai

- Minimizuoti [visą] atsakymo laiką:
  - Minimizuoti proceso apdorojimo laiką sistemoje (laikas pasiruošusių procesų eilėje, išskirti reikiamą CPU kvanto dydį)
- Maksimizuoti pralaidumą (angl. throughput):
  - Minimizuoti papildomas sąnaudas (pavyzdžiui proceso perjungimo konteksto skaičių)
  - Efektyviai naudoti esamus resursus (ne tik CPU, bet ir diskas, atmintis ir t.t.)

Planavimo politika - tikslai

- Minimizuoti [visą] atsakymo laiką:
  - Minimizuoti proceso apdorojimo laiką sistemoje (laikas pasiruošusių procesų eilėje, išskirti reikiamą CPU kvanto dydį)
- Maksimizuoti pralaidumą (angl. throughput):
  - Minimizuoti papildomas sąnaudas (pavyzdžiui proceso perjungimo konteksto skaičių)
  - Efektyviai naudoti esamus resursus (ne tik CPU, bet ir diskas, atmintis ir t.t.)
- Vykdyti sąžiningą resursų paskirstymą
  - Dalinti CPU resursus sąžiningai – subjektyvu, tačiau bendru atveju vertinamas bendras vartotojo pasitenkinimas sistemos darbu
  - Daugiau sąžiningumo – didesnis atsakymo laikas.

Planavimo įvertinimo kriterijai, metrika (1)

Kriterijaus pavadinimas	Kriterijaus aprašymas
Laukimo laikas (*waiting time*)	Laikas, kurį procesai praleidžia pasiruošusių procesų (*angl. ready queue*) eilėje
Vykdymo laikas (*service, execution time*)	Laikas, kurį procesas yra vykdomas CPU
Visas vykdymo laikas (*turnaround*)	Laikas, kurio reikia tam tikram procesui atlikti.
Atsakymo laikas (*response time*)	Laiko tarpas nuo to momento, kai užklausa buvo pateikta, iki tol, kol buvo gauti pirmieji atsakymai

Planavimo įvertinimo kriterijai, metrika (1)

Kriterijaus pavadinimas	Kriterijaus aprašymas
Laukimo laikas (*waiting time*)	Laikas, kurį procesai praleidžia pasiruošusių procesų (*angl. ready queue*) eilėje
Vykdymo laikas (*service, execution time*)	Laikas, kurį procesas yra vykdomas CPU
Visas vykdymo laikas (*turnaround*)	Laikas, kurio reikia tam tikram procesui atlikti.
Atsakymo laikas (*response time*)	Laiko tarpas nuo to momento, kai užklausa buvo pateikta, iki tol, kol buvo gauti pirmieji atsakymai

- Laikas, per kurį gauname sukompiliuotą programą

Planavimo įvertinimo kriterijai, metrika (1)

Kriterijaus pavadinimas	Kriterijaus aprašymas
Laukimo laikas (*waiting time*)	Laikas, kurį procesai praleidžia pasiruošusių procesų (*angl. ready queue*) eilėje
Vykdymo laikas (*service, execution time*)	Laikas, kurį procesas yra vykdomas CPU
Visas vykdymo laikas (*turnaround*)	Laikas, kurio reikia tam tikram procesui atlikti.
Atsakymo laikas (*response time*)	Laiko tarpas nuo to momento, kai užklausa buvo pateikta, iki tol, kol buvo gauti pirmieji atsakymai

- Laikas, per kurį gauname sukompiliuotą programą
- Laikas, per kurį sistema pateikia įvestus žodžius į ekraną

Planavimo įvertinimo kriterijai, metrika (1)

Kriterijaus pavadinimas	Kriterijaus aprašymas
Laukimo laikas (*waiting time*)	Laikas, kurį procesai praleidžia pasiruošusių procesų (*angl. ready queue*) eilėje
Vykdymo laikas (*service, execution time*)	Laikas, kurį procesas yra vykdomas CPU
Visas vykdymo laikas (*turnaround*)	Laikas, kurio reikia tam tikram procesui atlikti.
Atsakymo laikas (*response time*)	Laiko tarpas nuo to momento, kai užklausa buvo pateikta, iki tol, kol buvo gauti pirmieji atsakymai

- Laikas, per kurį gauname sukompiliuotą programą
- Laikas, per kurį sistema pateikia įvestus žodžius į ekraną

Visas vykd./atsak. laikas = Laukimo laikas + Vykd. arba 1 CPU kvanto įvykd. laikas

Planavimo įvertinimo kriterijai, metrika (1)

Kriterijaus pavadinimas	Kriterijaus aprašymas
Laukimo laikas (*waiting time*)	Laikas, kurį procesai praleidžia pasiruošusių procesų (*angl. ready queue*) eilėje
Vykdymo laikas (*service, execution time*)	Laikas, kurį procesas yra vykdomas CPU
Visas vykdymo laikas (*turnaround*)	Laikas, kurio reikia tam tikram procesui atlikti.
Atsakymo laikas (*response time*)	Laiko tarpas nuo to momento, kai užklausa buvo pateikta, iki tol, kol buvo gauti pirmieji atsakymai
Pralaidumas (*throughput*)	Skaičius procesų, kurie baigiami vykdyti per nurodytą laiko vienetą

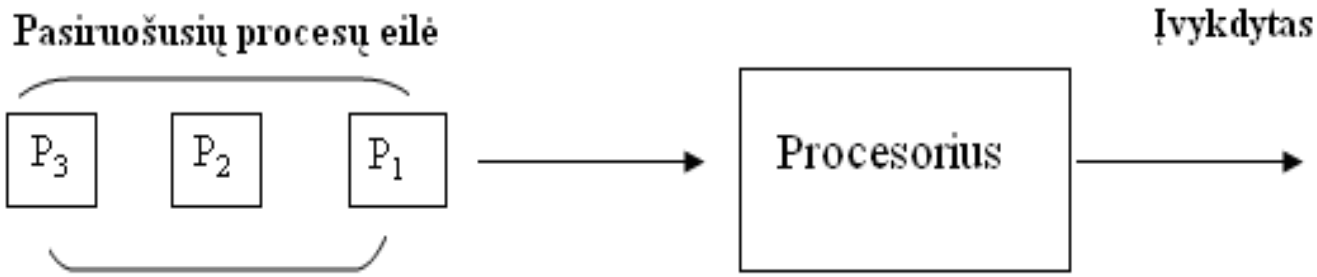
Planavimo įvertinimo kriterijai, metrika (1)

Kriterijaus pavadinimas	Kriterijaus aprašymas
Laukimo laikas (*waiting time*)	Laikas, kurį procesai praleidžia pasiruošusių procesų (*angl. ready queue*) eilėje
Vykdymo laikas (*service, execution time*)	Laikas, kurį procesas yra vykdomas CPU
Visas vykdymo laikas (*turnaround*)	Laikas, kurio reikia tam tikram procesui atlikti.
Atsakymo laikas (*response time*)	Laiko tarpas nuo to momento, kai užklausa buvo pateikta, iki tol, kol buvo gauti pirmieji atsakymai
Pralaidumas (*throughput*)	Skaičius procesų, kurie baigiami vykdyti per nurodytą laiko vienetą

•sąžiningumas •balansas •kritinių laiko ribų išlaikymas •nuspėjamumas

FCFS algoritmas

- pirmas pasirodęs procesas yra ir aptarnaujamas pirmas (FIFO), CPU perėmimas negalimas:
  - Ankstyvosiose OS procesas vykdomas iki jo pabaigos (tame tarpe ir su I/O)
  - Dabartinėse OS procesas vykdomas iki jo pabaigos arba užsiblokavimo



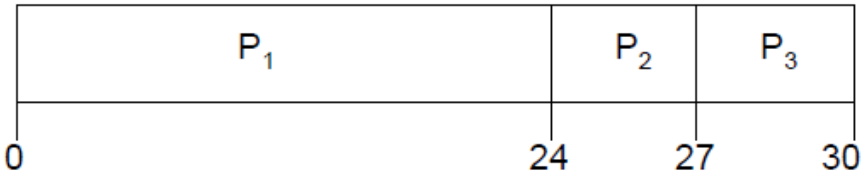
FCFS pavyzdys

Procesas	Vykdymo laikas
P1	24
P2	3
P3	3

FCFS pavyzdys

Procesas	Vykdymo laikas
P1	24
P2	3
P3	3

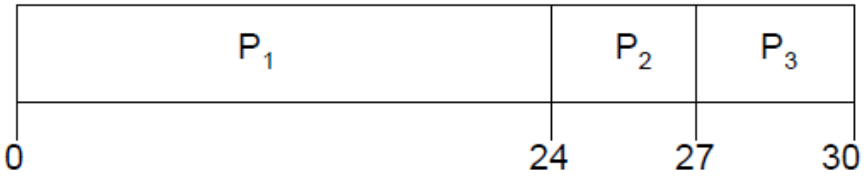
(a) Procesai pasirodo seka P1, P2, P3



FCFS pavyzdys

Procesas	Vykdymo laikas
P1	24
P2	3
P3	3

(a) Procesai pasirodo seka P1, P2, P3



proceso P1 laukimo eilėje laikas *WT1* 0

proceso P2 laukimo eilėje laikas *WT2* 24

proceso P3 laukimo eilėje laikas *WT3* 27

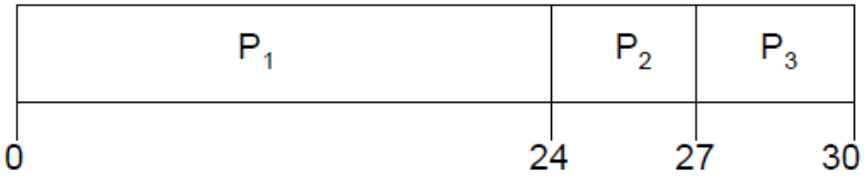
vidutinis laukimo eilėje laikas WT_{avg} (0+24+27)/3=17

vidutinis vykdymo/atsakymo laikas RT_{avg} (24+27+30)/3=27

FCFS pavyzdys

Procesas	Vykdymo laikas
P1	24
P2	3
P3	3

(a) Procesai pasirodo seka P1, P2, P3



proceso P1 laukimo eilėje laikas *WT1* — 0

proceso P2 laukimo eilėje laikas *WT2* — 24

proceso P3 laukimo eilėje laikas *WT3* — 27

vidutinis laukimo eilėje laikas WT_{avg} — (0+24+27)/3=17

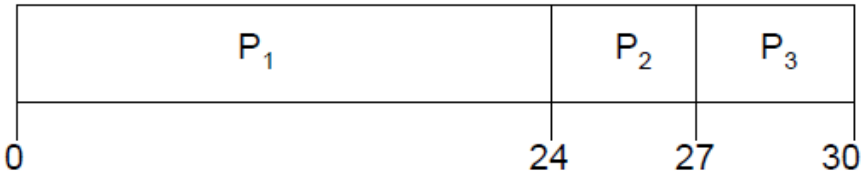
vidutinis vykdymo/atsakymo laikas RT_{avg} — (24+27+30)/3=27

Konvojaus efektas – visi procesai laukia vieno ilgo proceso pabaigos

FCFS pavyzdys

Procesas	Vykdymo laikas
P1	24
P2	3
P3	3

(a) Procesai pasirodo seka P1, P2, P3



(b) Procesai pasirodo seka P2, P3, P1

proceso P1 laukimo eilėje laikas *WT1* — 0

proceso P2 laukimo eilėje laikas *WT2* — 24

proceso P3 laukimo eilėje laikas *WT3* — 27

Skaičiuojam...

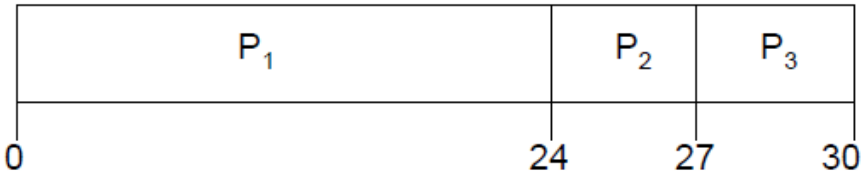
vidutinis laukimo eilėje laikas WT_{avg} — (0+24+27)/3=17

vidutinis vykdymo/atsakymo laikas RT_{avg} — (24+27+30)/3=27

FCFS pavyzdys

Procesas	Vykdymo laikas
P1	24
P2	3
P3	3

(a) Procesai pasirodo seka P1, P2, P3



(b) Procesai pasirodo seka P2, P3, P1

proceso P1 laukimo eilėje laikas *WT1*	0	6
proceso P2 laukimo eilėje laikas *WT2*	24	0
proceso P3 laukimo eilėje laikas *WT3*	27	3
vidutinis laukimo eilėje laikas WT_{avg}	(0+24+27)/3=17	(6+0+3)/3=3
vidutinis vykdymo/atsakymo laikas RT_{avg}	(24+27+30)/3=27	(3+6+30)/3=13

FSFC savybės

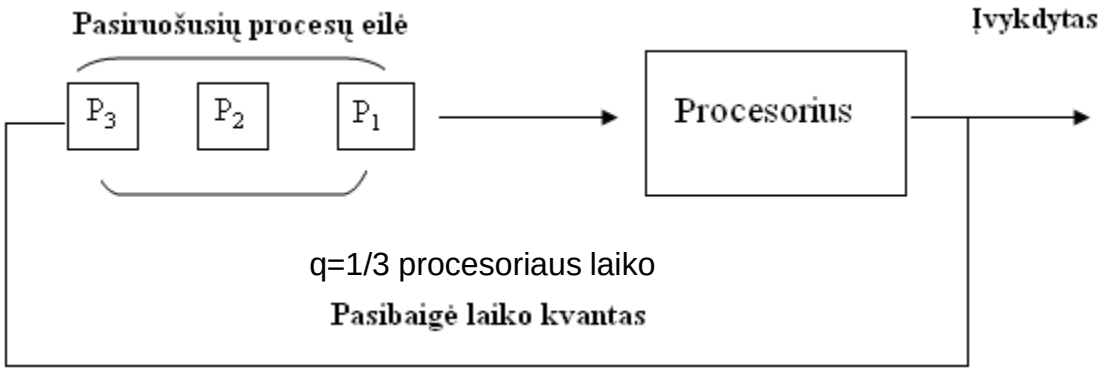
- Pastaba:
  - Gerai tinka daugiau **skaičiavimų** pobūdžio (CPU-bound) tipo procesams, ***nėra paranki*** trumpiems procesams.

- Problemos:
  - Vidutinis laukimo laikas gali būti didelis, jei trumpos užduotys laukia už ilgų (išsprendžiama)

- Privalumai:
  - Paprasta realizacija

RR algoritmas

- Sprendžia FCFS konvojaus efekto problemą.
- Vykdomas ciklinis aptarnavimas:
  - OS priskiria ***laiko kvantus*** (q) kiekvienam iš procesų
  - laiko kvantui pasibaigus, procesas pertraukiamas ir perkeliamas į pasiruošusių procesų eilės pabaigą laukti, kol jam bus išskirtas sekantis laiko kvantas.
  - n pasiruošusių procesų, kai laiko kvantas lygus q:
    - Kiekvienas procesas gauna 1/n CPU laiko
    - CPU laikas išdalinamas lygiomis dalimis q
    - Nei vienas procesas nelaukia daugiau kaip (n-1)q laiko

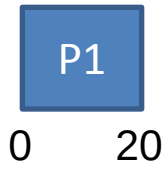


RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	**33**
P2	8	8
P3	68	68
P4	24	24

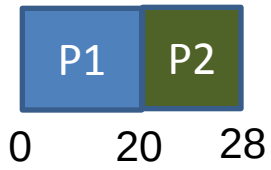
q = 20

Procesai pasirodo seka P1, P2, P3, P4



RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	33
P2	8	0
P3	68	68
P4	24	24



q = 20

Procesai pasirodo seka P1, P2, P3, P4

RR pavyzdys

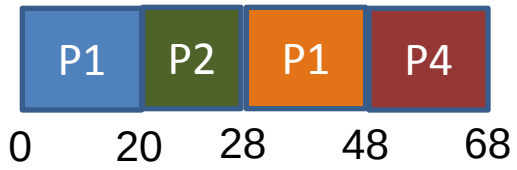
Procesas	Vykdymo laikas	Likęs laikas
P1	53	33
P2	8	0
P3	68	48
P4	24	24

q = 20

Procesai pasirodo seka P1, P2, P3, P4

RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	33
P2	8	0
P3	68	48
P4	24	4

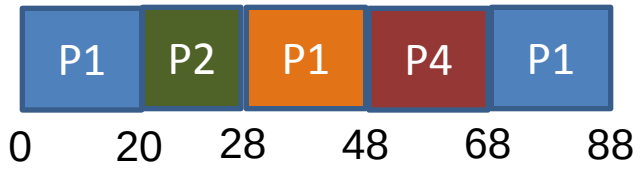


q = 20

Procesai pasirodo seka P1, P2, P3, P4

RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	13
P2	8	0
P3	68	48
P4	24	4



q = 20

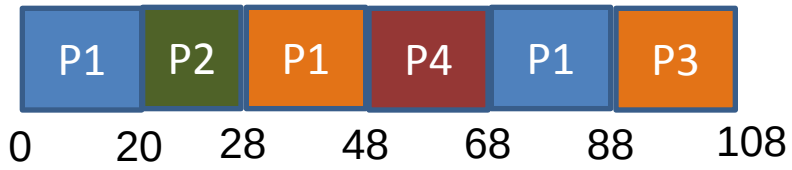
Procesai pasirodo seka P1, P2, P3, P4

RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	13
P2	8	0
P3	68	28
P4	24	4

q = 20

Procesai pasirodo seka P1, P2, P3, P4

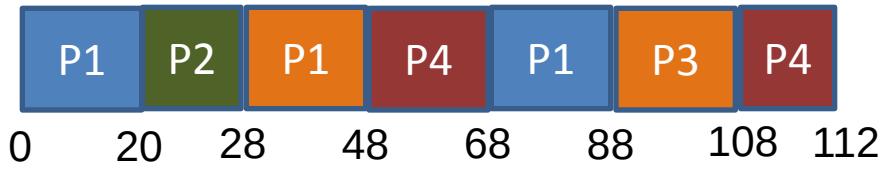


RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	13
P2	8	0
P3	68	28
P4	24	0

$q = 20$

Procesai pasirodo seka P1, P2, P3, P4

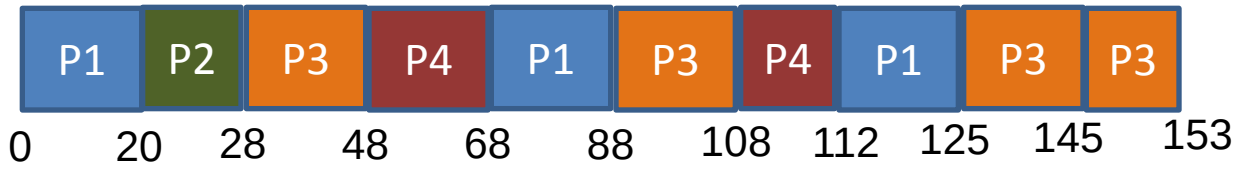


RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	0
P2	8	0
P3	68	0
P4	24	0

q = 20

Procesai pasirodo seka P1, P2, P3, P4

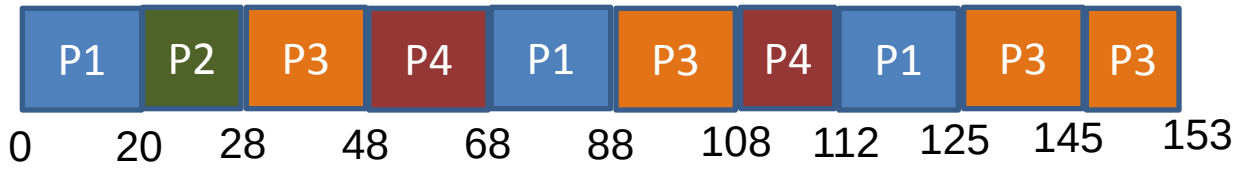


RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	0
P2	8	0
P3	68	0
P4	24	0

q = 20

Procesai pasirodo seka P1, P2, P3, P4



RR pavyzdys

Procesas	Vykdymo laikas	Likęs laikas
P1	53	0
P2	8	0
P3	68	0
P4	24	0

$q = 20$

Procesai pasirodo seka P1, P2, P3, P4

WT_{P1} = (68-20)+(112-88)= 72

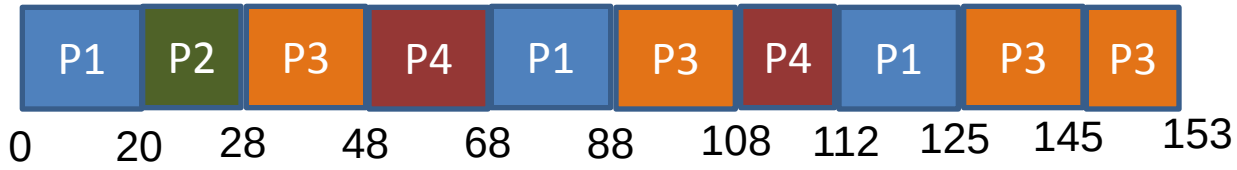
WT_{P2} = (20-0)= 20

WT_{P3} = (28-0)+(88-48)+(125-108)= 85

WT_{P4} = (48-0)+(108-68)= 88

WTavg = (72+20+85+88)/4=**66,25**

RTavg = (125+28+153+112)/4=**104,25**



RR savybės

- Privalumai:
  - Tinkamesnė esant trumpiems procesams
- Problemos:
  - Papildomas laikas dėl proceso konteksto perjungimo ilgai vykdomiems procesams (FCFS atveju papildomas laikas būtų mažesnis)
  - Laiko kvanto dydžio parinkimas:
    - Per didelis – sistemos atsakymo laikas netinkamas
    - Begalinis – FCFS atvejis
    - Per mažas – CPU pralaidumas mažėja
    - q optimalus, kai interaktyvūs procesai spėtų pateikti pateikti I/O užklausą (10-100ms), ~1 proc. papildomų sąnaudų dėl konteksto perjungimo)

FCFS ir RR palyginimas

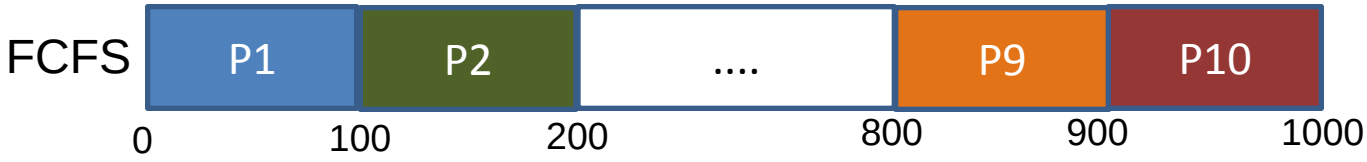
- Jei darome prielaidą, kad neturime konteksto perjungimo sąnaudų, ar visuomet RR bus efektyvesnė nei FCFS?

FCFS ir RR palyginimas

- Jei darome prielaidą, kad neturime konteksto perjungimo sąnaudų, ar visuomet RR bus efektyvesnė nei FCFS?
- Pavyzdys.
  - 10 procesų, kurių kiekvienam atlikti reikia 100 s CPU laiko.
  - Laiko kvantas q=1s.
  - Procesai pasirodo vienu metu.

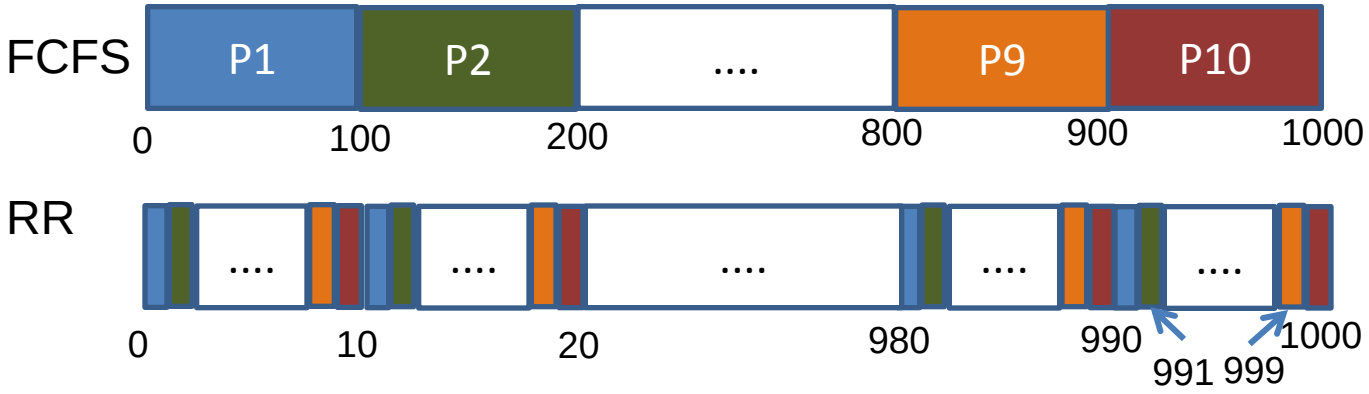
FCFS ir RR palyginimas

- Jei darome prielaidą, kad neturime konteksto perjungimo sąnaudų, ar visuomet RR bus efektyvesnė nei FCFS?
- Pavyzdys.
  - 10 procesų, kurių kiekvienam atlikti reikia 100 s CPU laiko.
  - Laiko kvantas q=1s.
  - Procesai pasirodo vienu metu.



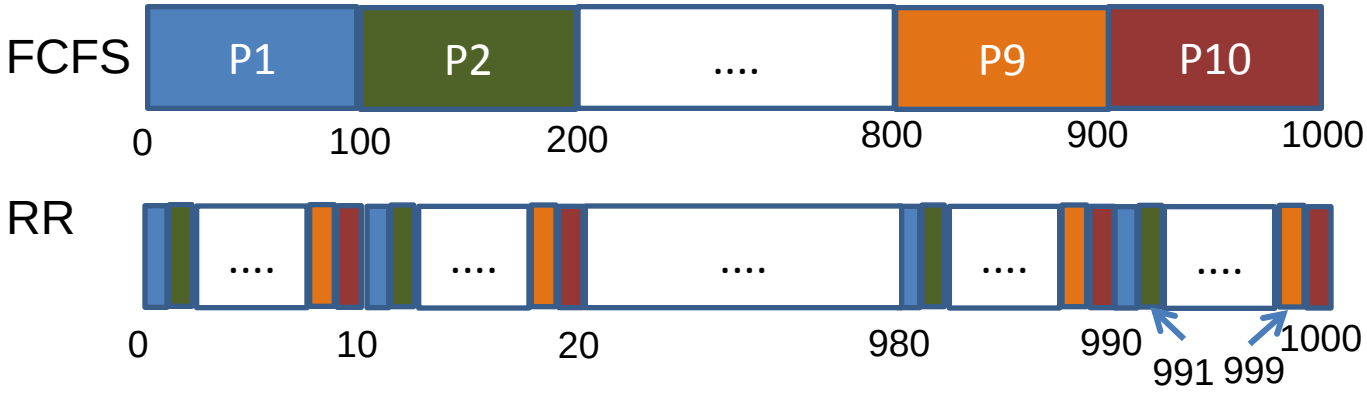
FCFS ir RR palyginimas

- Jei darome prielaidą, kad neturime konteksto perjungimo sąnaudų, ar visuomet RR bus efektyvesnė nei FCFS?
- Pavyzdys.
  - 10 procesų, kurių kiekvienam atlikti reikia 100 s CPU laiko.
  - Laiko kvantas q=1s.
  - Procesai pasirodo vienu metu.



FCFS ir RR palyginimas

- Jei darome prielaidą, kad neturime konteksto perjungimo sąnaudų, ar visuomet RR bus efektyvesnė nei FCFS?
- Pavyzdys.
  - 10 procesų, kurių kiekvienam atlikti reikia 100 s CPU laiko.
  - Laiko kvantas q=1s.
  - Procesai pasirodo vienu metu.

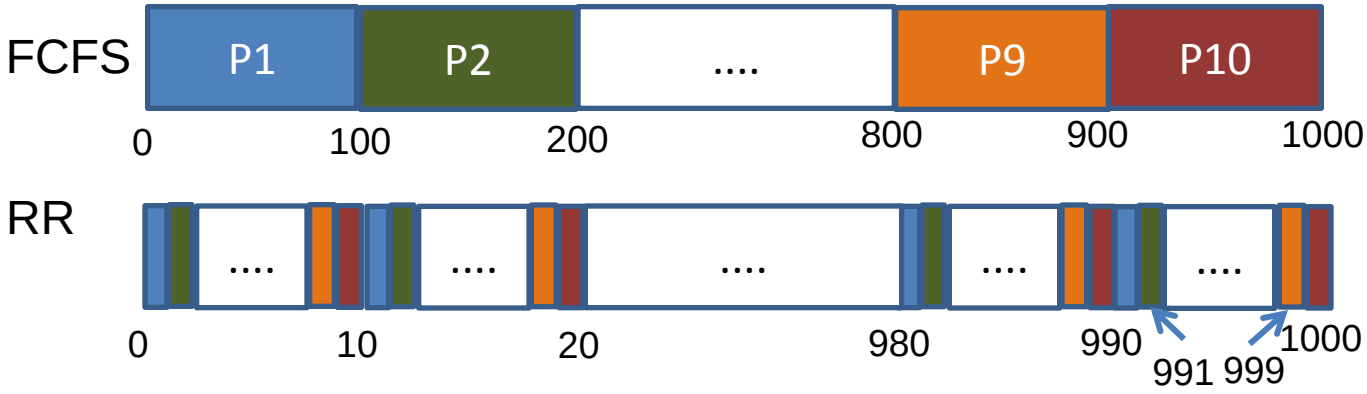


RTavg

Už. #	FCFS	RR
1	100	991
2	200	992
....	...	...
9	900	999
10	1000	1000

FCFS ir RR palyginimas (išvados)

- Užduotys tiek RR, tiek FCFS atveju baigiamos vykdyti tuo pačiu laiko momentu.
- Vidutinis atsakymo laikas RR atveju daug prastesnis
  - Prasta situacija, kai užduotims reikalingi vienodi/panašūs CPU resursai
- Spartinančioji atmintinė turi būti padalinta visiems procesams RR atveju, bet gali būti pilnai naudojama tik vieno proceso FCFS atveju
  - Bendras procesų vykdymo laikas RR atveju didesnis net nevertintant konteksto perjungimo sąnaudų



Už. #	FCFS	RR
1	100	991
2	200	992
....	...	...
9	900	999
10	1000	1000

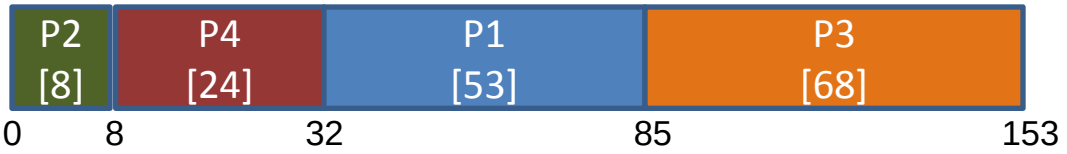
FCFS ir RR palyginimas esant skirtingiems laiko kvantams

P1 – 53, P2 – 8, P3 – 68, P4 – 24

	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT						
Vykdymo/ Atsakymo laikas RT						

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

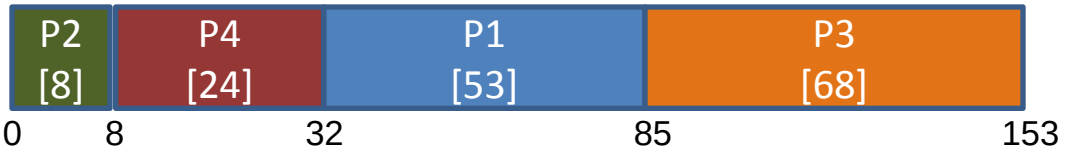
Geriausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT						
Vykdymo/ Atsakymo laikas RT						

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

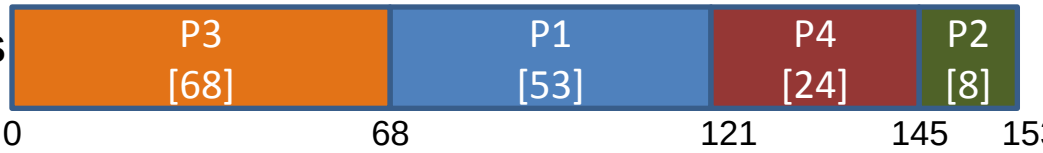
Geriausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

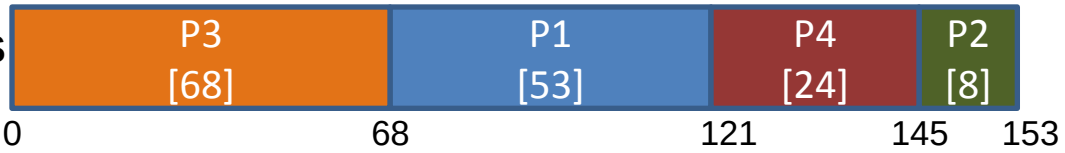
Blogiausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

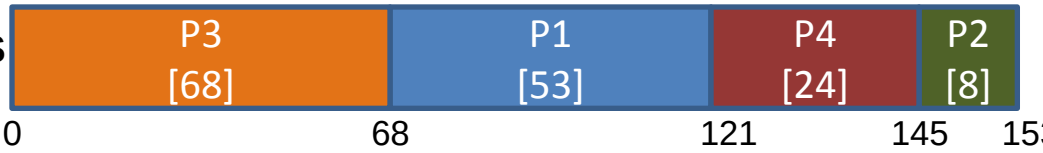
Blogiausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
	Pesim. FCFS	68	145	0	121	83,5
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5
	Pesim. FCFS	121	153	68	145	121,75

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

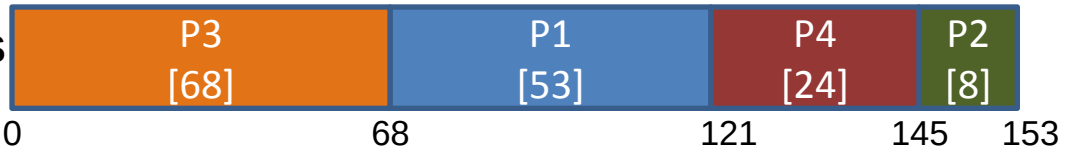
Blogiausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
	Q=8	80	8	85	56	57,25
	Pesim. FCFS	68	145	0	121	83,5
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5
	Q=8	133	16	153	80	95,5
	Pesim. FCFS	121	153	68	145	121,75

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

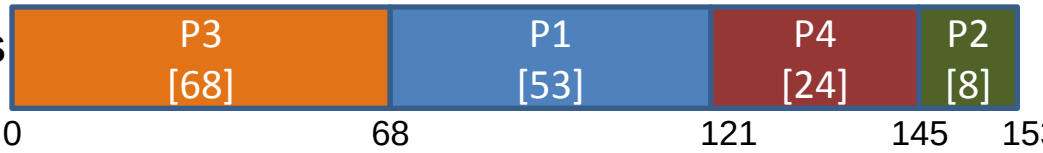
Blogiausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
	Q=1	84	22	85	57	**62**
	Q=8	80	8	85	56	57,25
	Q=20	72	20	85	88	**66,25**
	Pesim. FCFS	68	145	0	121	83,5
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5
	Q=1	137	30	153	81	**100,5**
	Q=8	133	16	153	80	95,5
	Q=20	125	28	153	112	**104,5**
	Pesim. FCFS	121	153	68	145	121,75

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

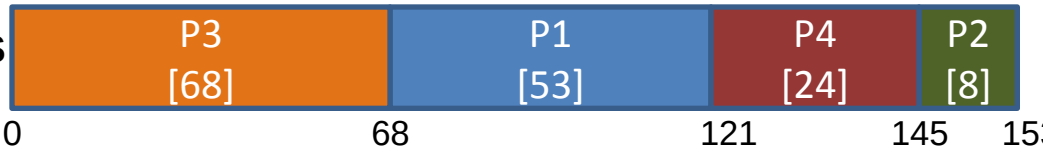
Blogiausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
	Q=1	84	22	85	57	62
	Q=5	82	20	85	68	**61,25**
	Q=8	80	8	85	56	57,25
	Q=10	82	10	85	68	**61,25**
	Q=20	72	20	85	88	66,25
	Pesim. FCFS	68	145	0	121	83,5
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5
	Q=1	137	30	153	81	100,5
	Q=5	135	28	153	82	**99,5**
	Q=8	133	16	153	80	95,5
	Q=10	135	18	153	92	**99,5**
	Q=20	125	28	153	112	104,5
	Pesim. FCFS	121	153	68	145	121,75

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

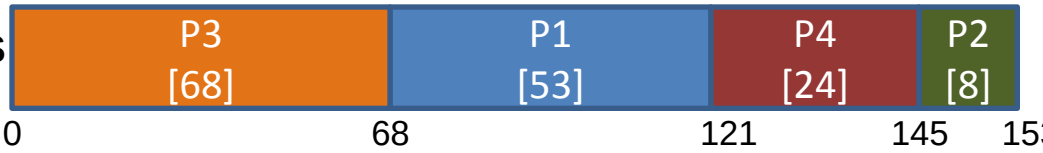
Blogiausias var. FCFS



	Laiko kvantas	**P1**	**P2**	**P3**	**P4**	**avg**
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
	Q=1	84	22	85	57	62
	Q=5	82	20	85	68	**61,25**
	Q=8	80	8	85	56	57,25
	Q=10	82	10	85	68	**61,25**
	Q=20	72	20	85	88	66,25
	Pesim. FCFS	68	145	0	121	83,5
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5
	Q=1	137	30	153	81	100,5
	Q=5	135	28	153	82	**99,5**
	Q=8	133	16	153	80	95,5
	Q=10	135	18	153	92	**99,5**
	Q=20	125	28	153	112	104,5
	Pesim. FCFS	121	153	68	145	121,75

FCFS ir RR palyginimas esant skirtingiems laiko kvantams

Blogiausias var. FCFS



	Laiko kvantas	P1	P2	P3	P4	avg
Laukimo eilėje laikas WT	Optim. FCFS	32	0	85	8	31,25
	Q=1	84	22	85	57	62
	Q=5	82	20	85	68	**61,25**
	Q=8	80	8	85	56	57,25
	Q=10	82	10	85	68	**61,25**
	Q=20	72	20	85	88	66,25
	Pesim. FCFS	68	145	0	121	83,5
Vykdymo/ Atsakymo laikas RT	Optim. FCFS	85	8	153	32	69,5
	Q=1	137	30	153	81	100,5
	Q=5	135	28	153	82	**99,5**
	Q=8	133	16	153	80	95,5
	Q=10	135	18	153	92	**99,5**
	Q=20	125	28	153	112	104,5
	Pesim. FCFS	121	153	68	145	121,75

O jei žinotume ateitį?

- Iš anksto žinodami procesų įvykdymo laikus, galėtume turėti optimistinį FCFS variantą.
- Trumpiausias procesas pirmas (**SJF** - Shortest Job First):
  - aptarnavimui yra renkamas procesas, kurio laukiamas įvykdymo laikas yra mažiausias.
  - reikalauja įvertinti, kiek laiko reikės proceso įvykdymui.
  - algoritmas yra ***nepertraukiamo*** tipo.

O jei žinotume ateitį?

- Iš anksto žinodami procesų įvykdymo laikus, galėtume turėti optimistinį FCFS variantą.
- Trumpiausias procesas pirmas (**SJF** - Shortest Job First):
  - aptarnavimui yra renkamas procesas, kurio laukiamas įvykdymo laikas yra mažiausias.
  - reikalauja įvertinti, kiek laiko reikės proceso įvykdymui.
  - algoritmas yra ***nepertraukiamo*** tipo.
- Procesas, kuriam liko mažiausiai aptarnavimo laiko, aptarnaujamas pirmas (**SRTF** – Shortest Remaining Time First):
  - Aptarnavimui parenkamas procesas, įvertinant likusį aptarnavimo laiką
  - algoritmas yra ***pertraukiamo*** tipo.
  - reikalauja įvertinti likusį reikalaujamą CPU laiką

SJF ir SRTF tikslai

- Kuo greičiau įvykdyti trumpus procesus
- Pateikia geresnius vidutinius laukimo eilėje laikus duotam procesų rinkiniui

- Duoda didelį efektą trumpiems, ir mažą - ilgiems procesams.

- SJF arba SRTF – geriausias sprendimas, kai planavime kritinis parametras yra vidutinis įvykdymo laikas *RT*.

SRTF, FCFS ir RR palyginimas

- Jei procesų įvykdymo laikai vienodi

SRTF, FCFS ir RR palyginimas

- Jei procesų įvykdymo laikai vienodi
  - FCFS. (SRTF šiomis sąlygomis taip pat tampa FCFS)

SRTF, FCFS ir RR palyginimas

- Jei procesų įvykdymo laikai vienodi
  - FCFS. (SRTF šiomis sąlygomis taip pat tampa FCFS)

- Jei procesų įvykdymo laikai skirtingi?

SRTF, FCFS ir RR palyginimas

- Jei procesų įvykdymo laikai vienodi
  - FCFS. (SRTF šiomis sąlygomis taip pat tampa FCFS)

- Jei procesų įvykdymo laikai skirtingi?
  - SRTF (arba RR), nes trumpų procesų vykdymo neturėtų įtakoti pasiruošusių procesų eilėje stovintys ilgus įvykdymo laikus turintys procesai

SRTF, FCFS ir RR palyginimas

- Tarkime turime tris procesus:
  - A ir B – skaičiavimo procesai (CPU bounded, 1 savaitės trukmės)
  - C – interaktyvus procesas (I/O bounded, 1ms CPU, 9ms I/O)

SRTF, FCFS ir RR palyginimas

- Tarkime turime tris procesus:
  - A ir B – skaičiavimo procesai (CPU bounded, 1 savaitės trukmės)
  - C – interaktyvus procesas (I/O bounded, 1ms CPU, 9ms I/O)
- Jei apibrėžtu laiko tarpu turime tik vieną procesą, tai:
  - C atveju – 90% naudojamas kietasis diskas.
  - A arba B atveju – 100% naudojami CPU resursai.

SRTF, FCFS ir RR palyginimas

- Tarkime turime tris procesus:
  - A ir B – skaičiavimo procesai (CPU bounded, 1 savaitės trukmės)
  - C – interaktyvus procesas (I/O bounded, 1ms CPU, 9ms I/O)
- Jei apibrėžtu laiko tarpu turime tik vieną procesą, tai:
  - C atveju – 90% naudojamas kietasis diskas.
  - A arba B atveju – 100% naudojami CPU resursai.
- FCFS atveju –

SRTF, FCFS ir RR palyginimas

- Tarkime turime tris procesus:
  - A ir B – skaičiavimo procesai (CPU bounded, 1 savaitės trukmės)
  - C – interaktyvus procesas (I/O bounded, 1ms CPU, 9ms I/O)
- Jei apibrėžtu laiko tarpu turime tik vieną procesą, tai:
  - C atveju – 90% naudojamas kietasis diskas.
  - A arba B atveju – 100% naudojami CPU resursai.

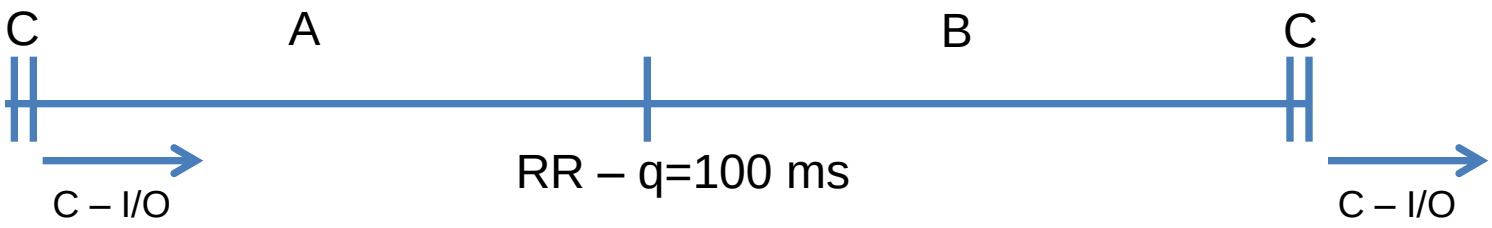
- FCFS atveju – A arba B gavę CPU, jį naudos savaitę

SRTF, FCFS ir RR palyginimas

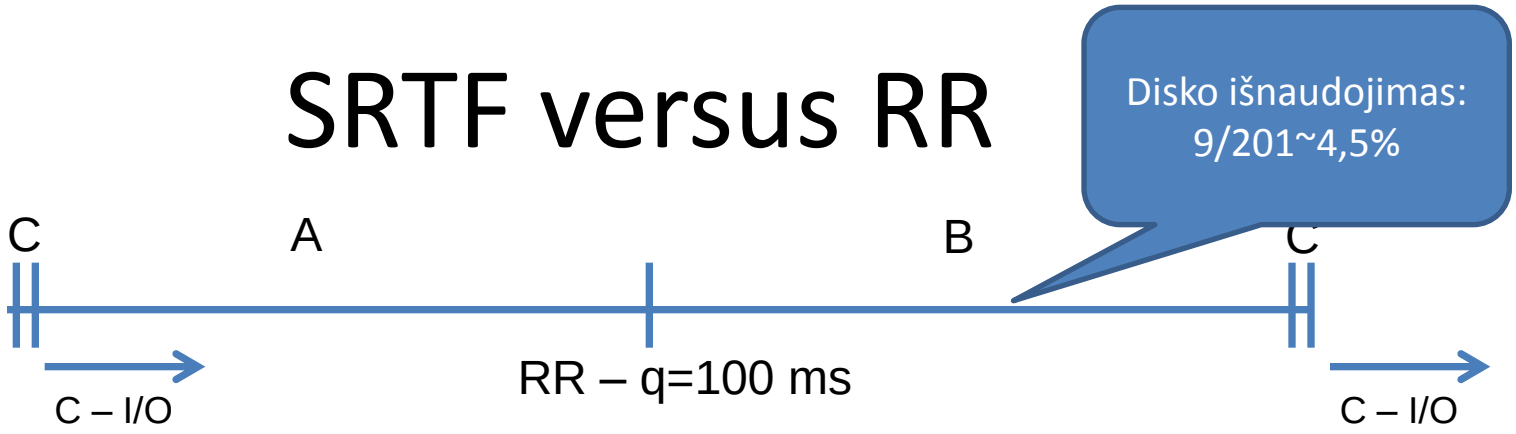
- Tarkime turime tris procesus:
  - A ir B – skaičiavimo procesai (CPU bounded, 1 savaitės trukmės)
  - C – interaktyvus procesas (I/O bounded, 1ms CPU, 9ms I/O)
- Jei apibrėžtu laiko tarpu turime tik vieną procesą, tai:
  - C atveju – 90% naudojamas kietasis diskas.
  - A arba B atveju – 100% naudojami CPU resursai.

- FCFS atveju – A arba B gavę CPU, jį naudos savaitę
- RR arba SRTF atveju –

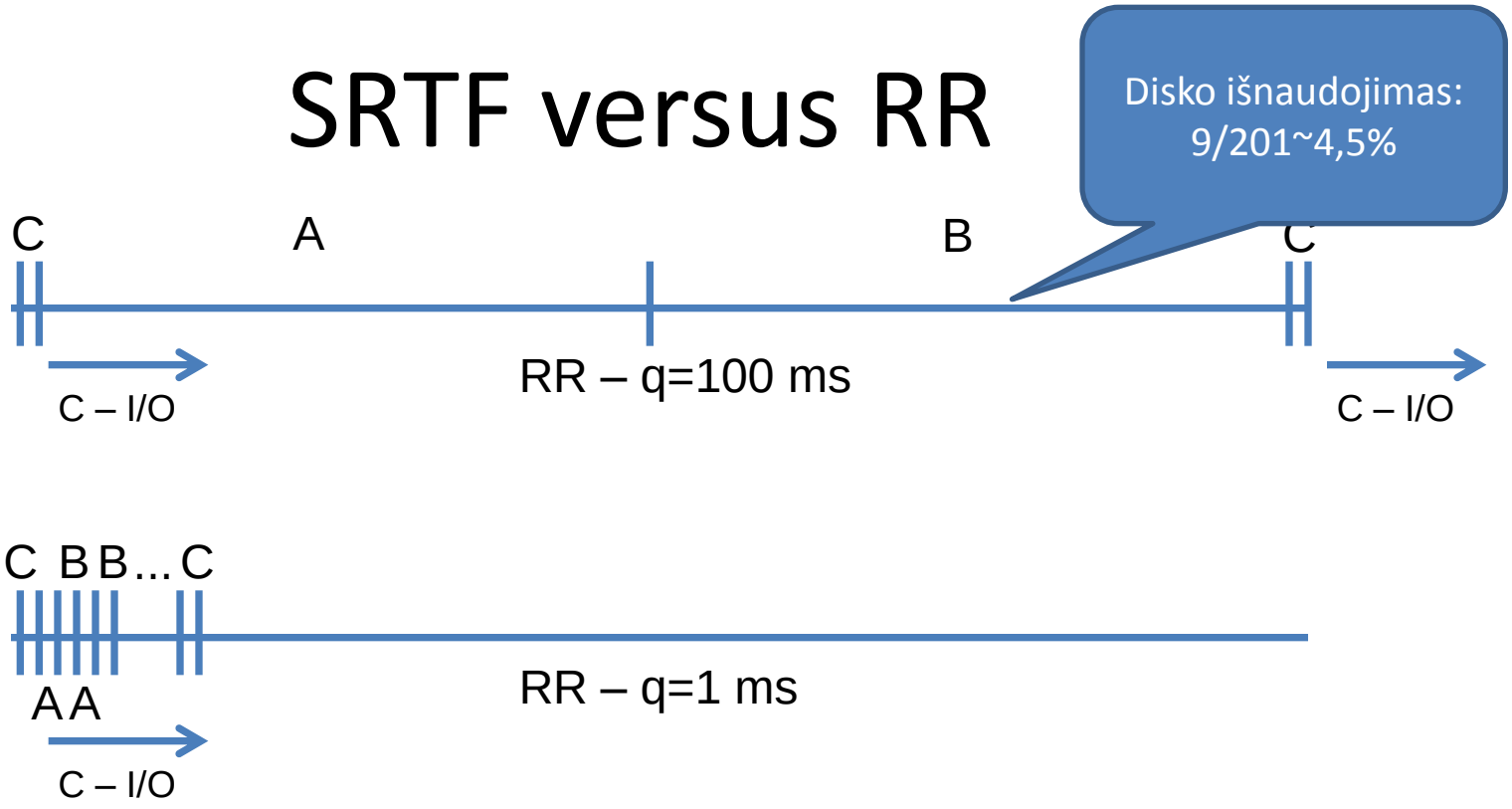
SRTF versus RR



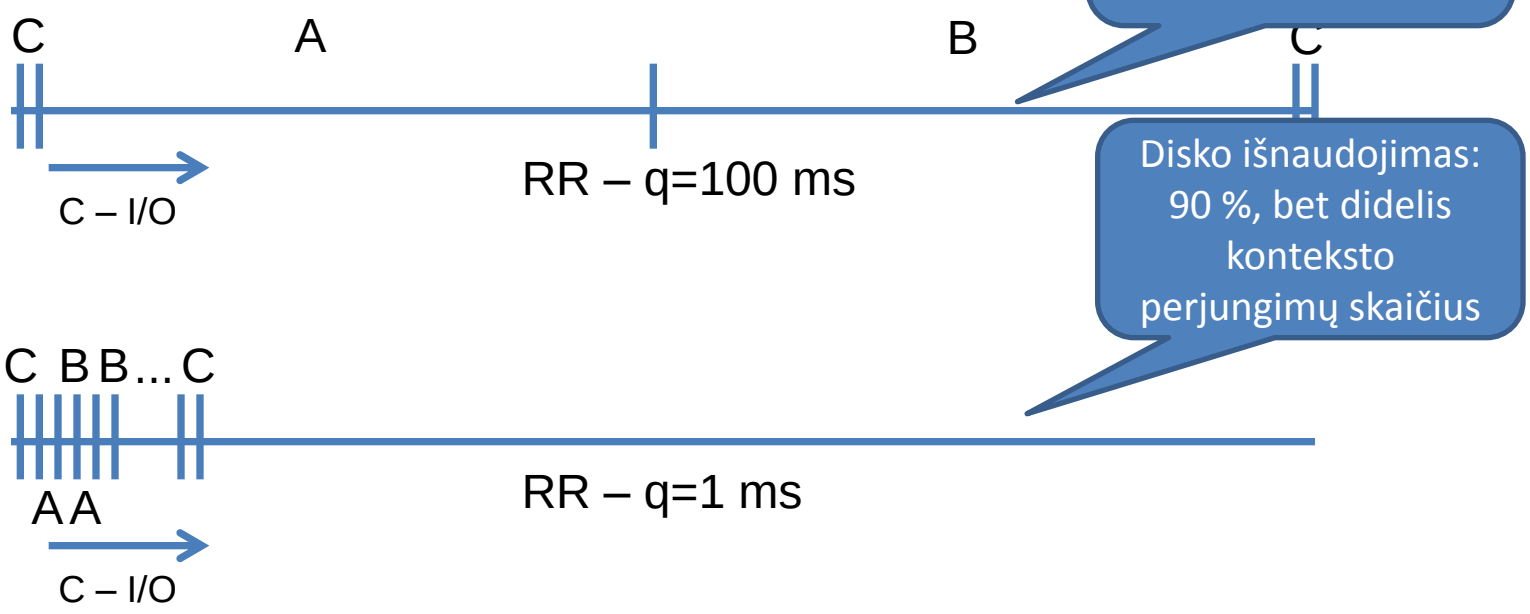
SRTF versus RR



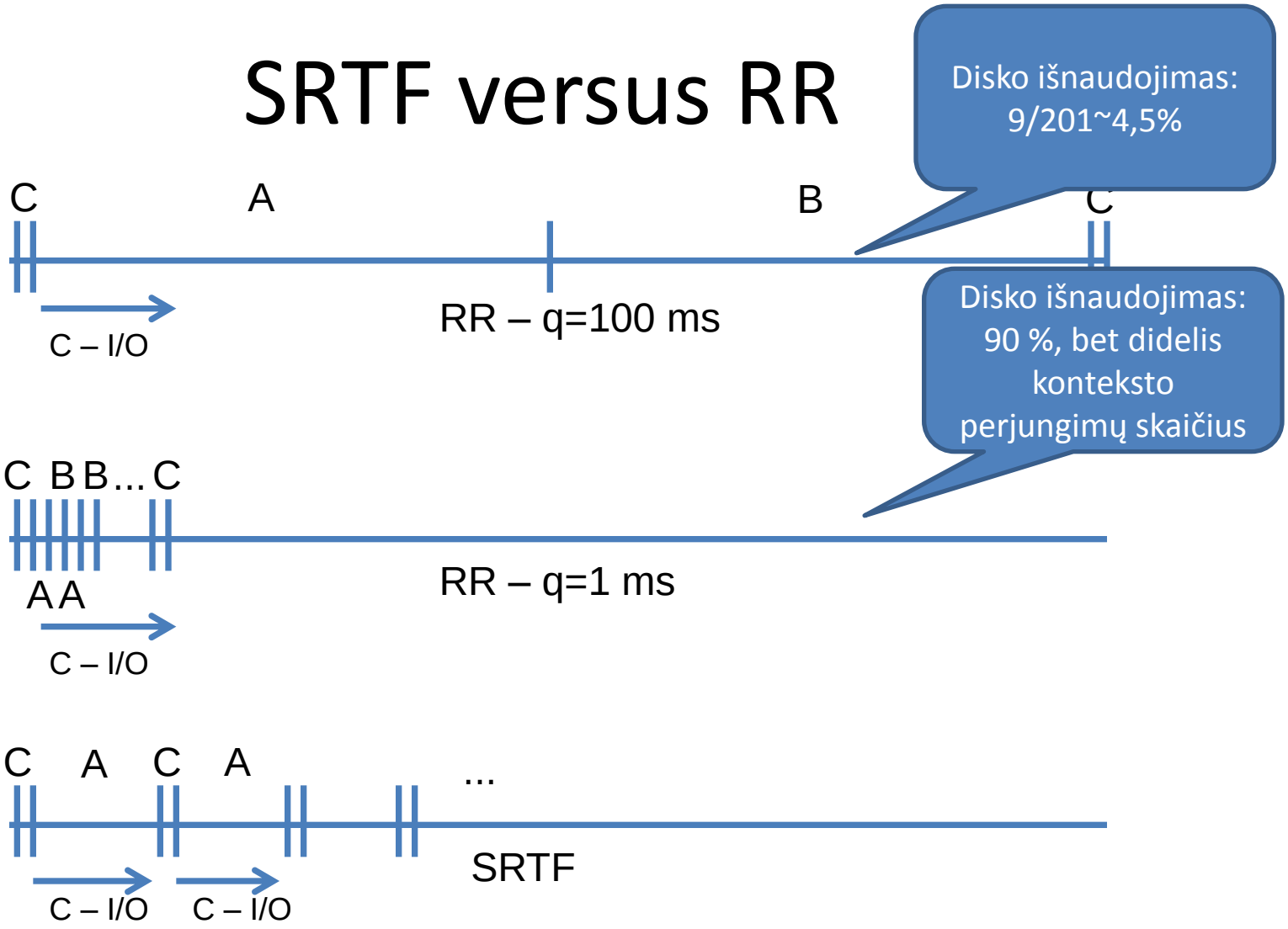
SRTF versus RR



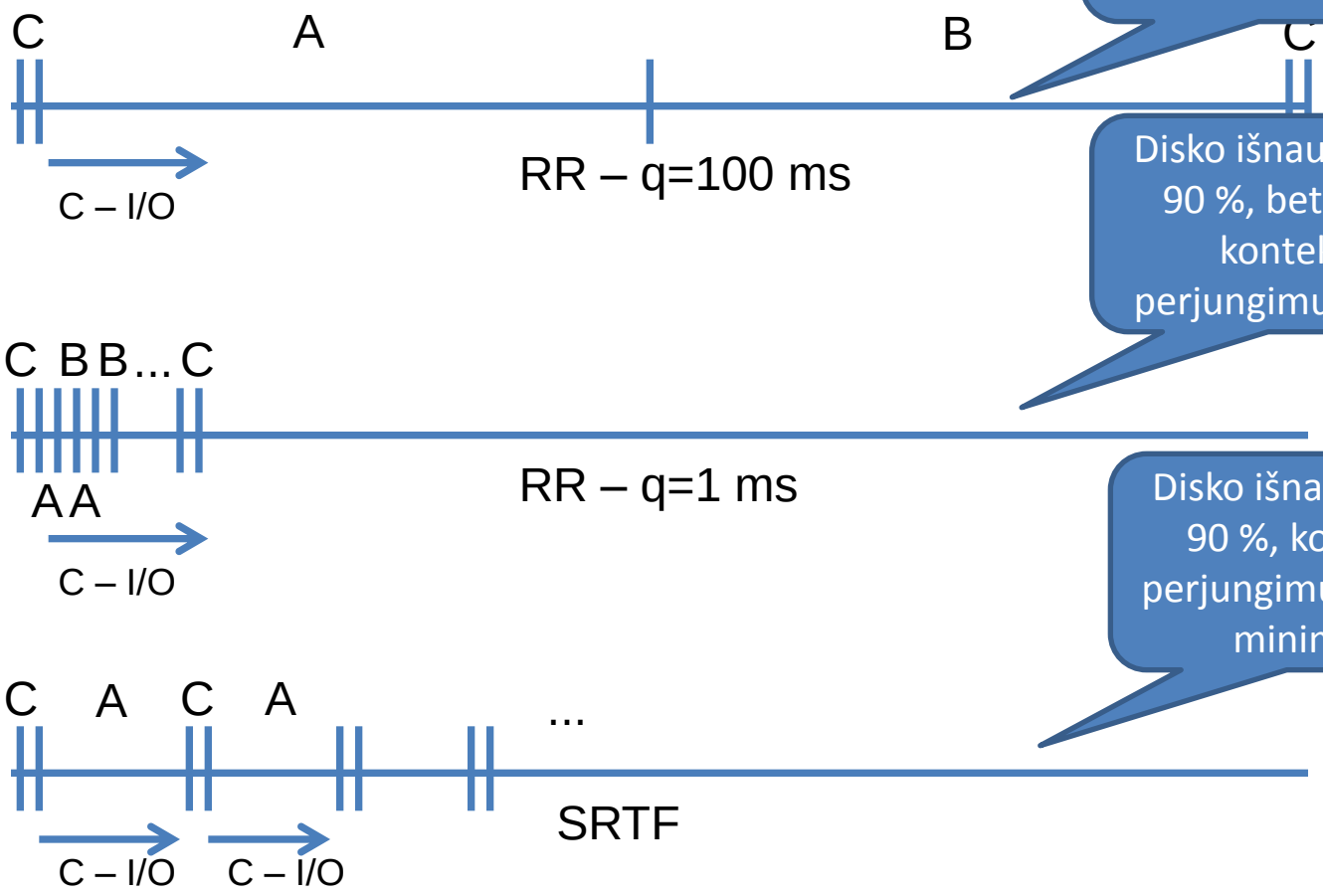
SRTF versus RR



SRTF versus RR



SRTF versus RR



Disko išnaudojimas: 9/201~4,5%

Disko išnaudojimas: 90 %, bet didelis konteksto perjungimų skaičius

Disko išnaudojimas: 90 %, konteksto perjungimų skaičius - minimalus

SRTF trūkumai

- Jei pasiruošusių procesų eilėje nuolat atsiranda trumpus vykdymo laikus turinčių procesų, ilgi procesai gali būti niekad nevykdomi – t.y. ***ilgesni procesai gali "badauti".***

SRTF trūkumai

- Jei pasiruošusių procesų eilėje nuolat atsiranda trumpus vykdymo laikus turinčių procesų, ilgi procesai gali būti niekad nevykdomi – t.y. ***ilgesni procesai gali "badauti".***

- CPU resursų paskirstymas procesams nesiremia sąžiningumo principu

SRTF trūkumai

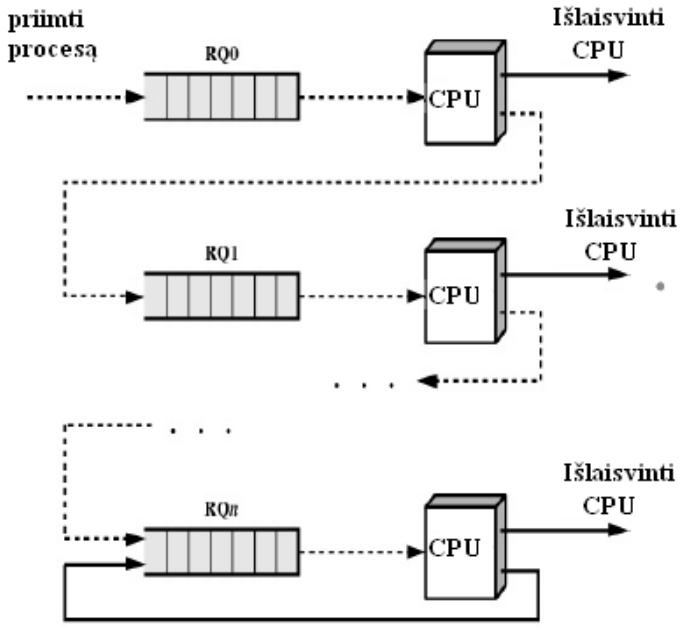
- Jei pasiruošusių procesų eilėje nuolat atsiranda trumpus vykdymo laikus turinčių procesų, ilgi procesai gali būti niekad nevykdomi – t.y. ***ilgesni procesai gali "badauti".***

- CPU resursų paskirstymas procesams nesiremia sąžiningumo principu

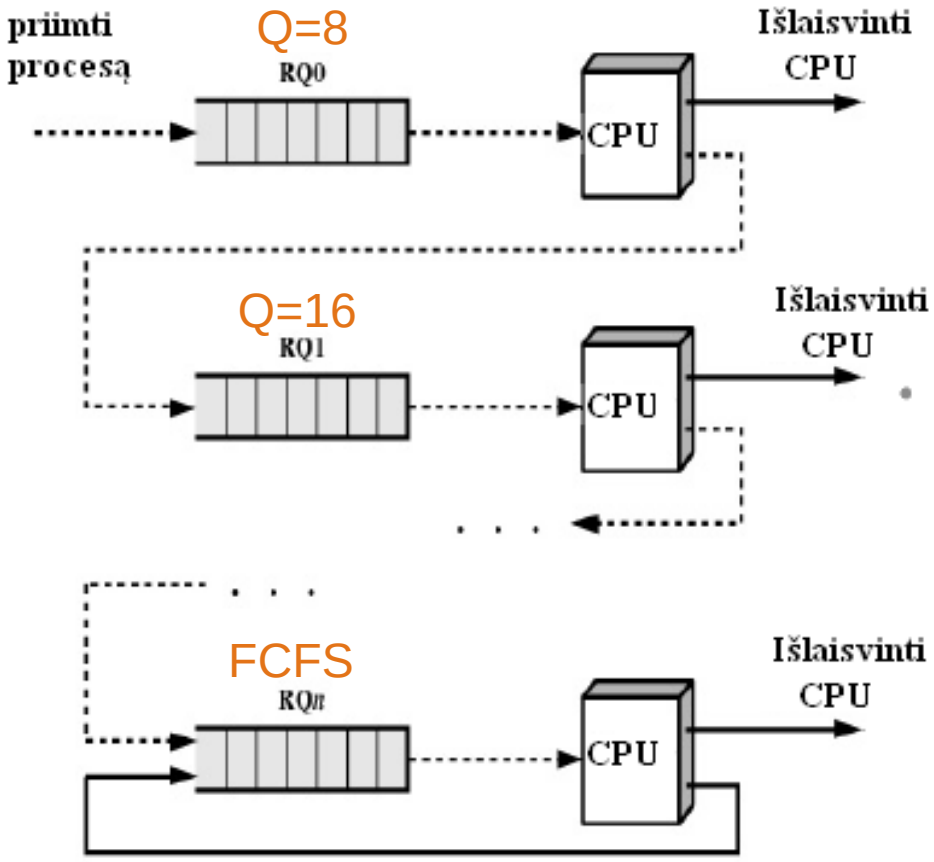
- Reikalinga įvertinti likusį reikalaujamą CPU laiką:
  - vartotojo pateiktas procesui reikalingas laikas.
  - Atliekama automatiškai (naudojant eksponentinį vidurkių skaičiavimą, kitus matematinius mechanizmus).

MLFQ

- Daugelio lygių grįžtamojo ryšio eilės (MLFQ, Multi-level feedback queues):
  - Pasiruošusių procesų eilė gali būti sudaloma į atskiras eiles, į jas talpinami skirtingo tipo procesai , eilės turi prioritetą.
  - Kiekviena eilė gali turėti savą procesų planavimo algoritmą.
  - Taikomas tam tikras mechanizmas planavimui tarp eilių



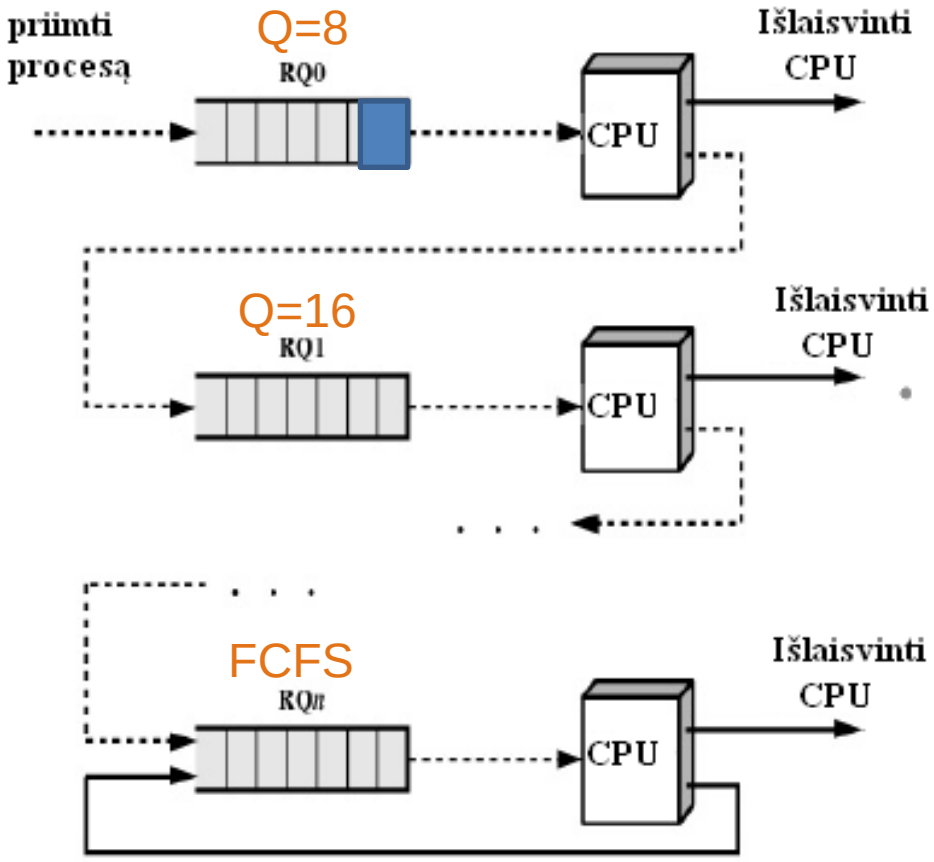
MLFQ



Pagr. veikimo principai:

- Pirmą kartą procesas talpinamas į aukščiausio prioriteto eilę.
- Jei išnaudotas CPU laiko kvantas, bet procesas nepasibaigė – perkelti jį į žemesnio prioriteto eilę
- Jei procesas neišnaudojo jam skirto laiko kvanto – perkelti jį į aukštesnio prioriteto eilę arba palikti esamoje.

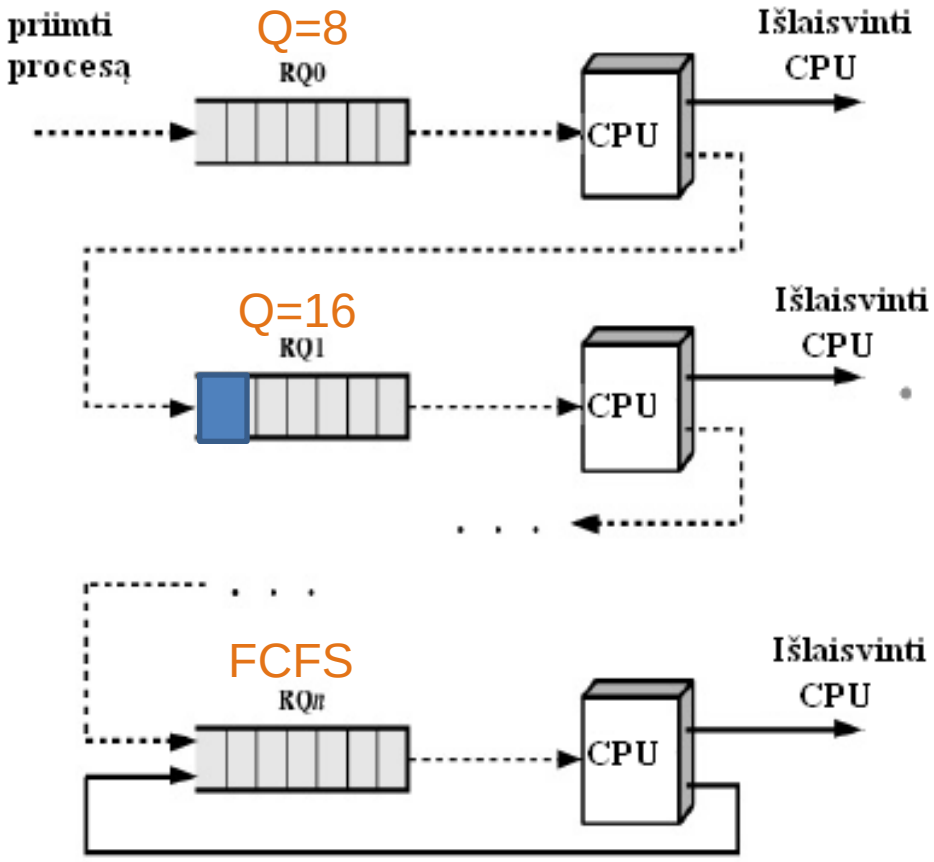
MLFQ



Pagr. veikimo principai:

- Pirmą kartą procesas talpinamas į aukščiausio prioriteto eilę.
- Jei išnaudotas CPU laiko kvantas, bet procesas nepasibaigė – perkelti jį į žemesnio prioriteto eilę
- Jei procesas neišnaudojo jam skirto laiko kvanto – perkelti jį į aukštesnio prioriteto eilę arba palikti esamoje.

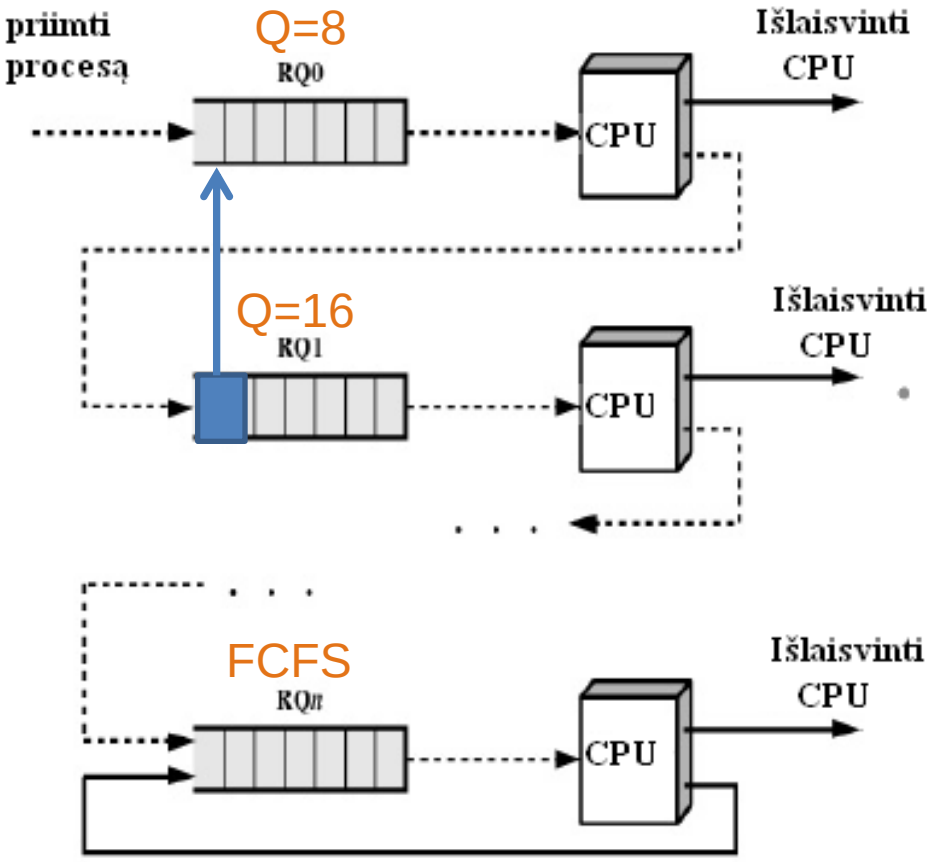
MLFQ



Pagr. veikimo principai:

- Pirmą kartą procesas talpinamas į aukščiausio prioriteto eilę.
- Jei išnaudotas CPU laiko kvantas, bet procesas nepasibaigė – perkelti jį į žemesnio prioriteto eilę
- Jei procesas neišnaudojo jam skirto laiko kvanto – perkelti jį į aukštesnio prioriteto eilę arba palikti esamoje.

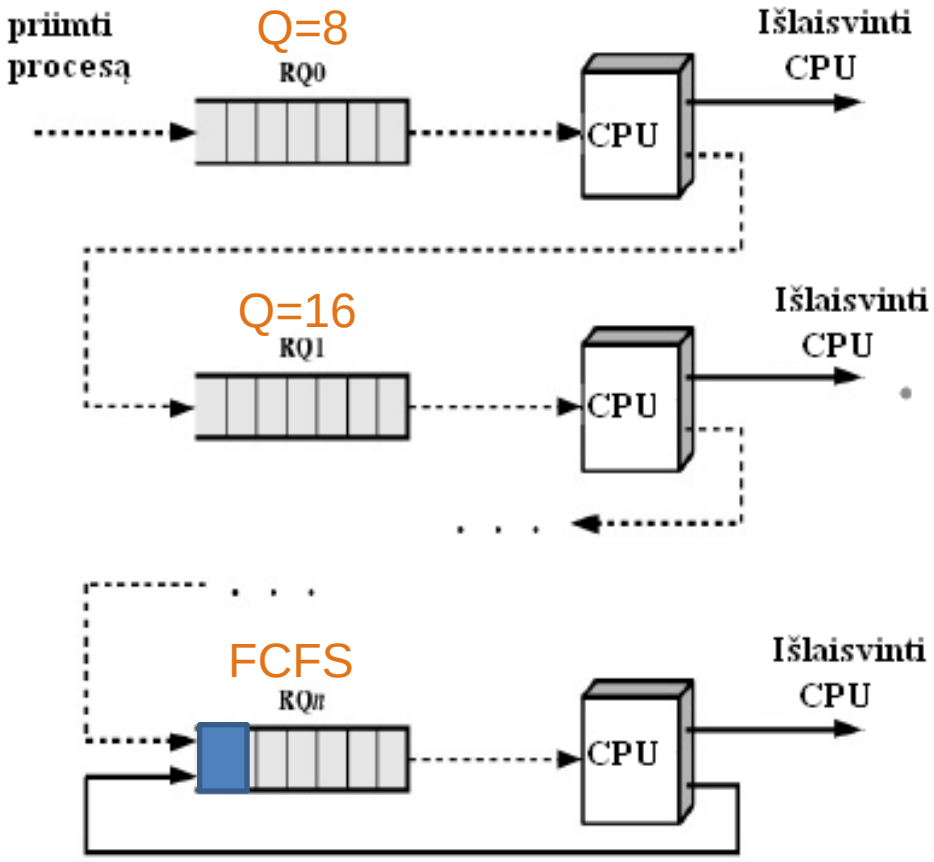
MLFQ



Pagr. veikimo principai:

- Pirmą kartą procesas talpinamas į aukščiausio prioriteto eilę.
- Jei išnaudotas CPU laiko kvantas, bet procesas nepasibaigė – perkelti jį į žemesnio prioriteto eilę
- Jei procesas neišnaudojo jam skirto laiko kvanto – perkelti jį į aukštesnio prioriteto eilę arba palikti esamoje.

MLFQ



Pagr. veikimo principai:

- Pirmą kartą procesas talpinamas į aukščiausio prioriteto eilę.
- Jei išnaudotas CPU laiko kvantas, bet procesas nepasibaigė – perkelti jį į žemesnio prioriteto eilę
- Jei procesas neišnaudojo jam skirto laiko kvanto – perkelti jį į aukštesnio prioriteto eilę arba palikti esamoje.

MLFQ

- Planavimas tarp eilių
  - Jis gali būti **fiksuotas**, pavyzdžiui, aptarnauti pradžioje visus interaktyvius procesus, tada foninius.
    - Ilgus vykdymo laikus turintys procesai gali iš vis negauti CPU resursų – t.y. galimas badavimas
    - Multics sistemoje, ją išjungiant rastas 10 metų senumo procesas.
  - Kitas sprendimas – kiekvienai **eilei taikyti laiko kvantavimo principą** – kiekviena eilė gauna dalį CPU laiko ir padalina jį eilėje esantiems procesams.
    - Pavyzdžiui, 80% skiriama interaktyvių procesų eilei, jai taikant RR discipliną, bei 20% foninių procesų eilei, jai taikant FCFS discipliną.

Sąžiningas planavimas

- Reikalingas kompromisas: sąžiningumas didina atsakymo laiką

Sąžiningas planavimas

- Reikalingas kompromisas: sąžiningumas didina atsakymo laiką
- Sąžiningumo mechanizmo realizacija:
  - kiekvienai eilei paskirti fiksuotą dalį CPU laiko → ne visuomet pasiteisina (kas atsitiks, jei turim 100 trumpų procesų ir vieną ilgą?)

Sąžiningas planavimas

- Reikalingas kompromisas: sąžiningumas didina atsakymo laiką
- Sąžiningumo mechanizmo realizacija:
  - kiekvienai eilei paskirti fiksuotą dalį CPU laiko → ne visuomet pasiteisina (kas atsitiks, jei turim 100 trumpų procesų ir vieną ilgą?)
  - Didinti prioritetą tiems procesams, kurie negauna CPU laiko (naudojama UNIX tipo OS) → neaišku kaip dažnai ir iki kokio prioriteto reikia didinti, kad būtų optimalu.

Solaris OS prioritetinės klasės

Solaris gali atpažinti 170 skirtingų prioritetų, 0-169.

Prioritetinės klasės:

- **Laiko dalinimo TS (timeshare)**: 0-59 prioritetas . Tai klasė skirta procesams bei juos atitinkančioms branduolio gijoms pagal nutylėjimą.
- **Interaktyvaus aptarnavimo IA (interactive)**: 0-59 prioritetas. Tai išplėsta TS klasės versija, siekiant skirti papildomus išteklius procesams, susijusiems su specifiniu langu.
- **Grupinio aptarnavimo FSS (fair-share scheduler)**: 0-59pr. Atitinkamos gijos aptarnaujamos, priklausomai nuo jos priklausymo tam tikrai grupei bei nuo CPU panaudojimo.
- **Fiksuoto prioriteto FX (fixed-priority)**: 0-59pr. Gijos turi fiksuotą prioritetą visu jų gyvavimo metu.

- **Sisteminė klasė SYS (system)**: 60-99 pr. tai branduolio gijų prioritetai. Šio tipo gijos vykdomos kol užsiblokuoja arba kol įvykdo visus veiksmus.

- **Realaus laiko klasė RT (real-time)**: 100-159 pr. RT gija gali perimti SYS tipo gijos veiksmus. Gijoms priskiriamas fiksuotas prioritetas ir fiksuotas laiko kvantas

Pasižiūrėti kokios klasės sukonfigūruotos: `dispadmin -l`

TS prioritetinės klasės dispečerio konfigūracija

```
-bash-3.00$ dispadmin -c TS -g
# Time Sharing Dispatcher Configuration
RES=1000

# ts_quantum  ts_tqexp  ts_slpret  ts_maxwait ts_lwait  PR- LEVEL
      200         0         50          0         50     #     0
      200         0         50          0         50     #     1
      200         0         50          0         50     #     2
      200         0         50          0         50     #     9
      160         0         51          0         51     #    10
      160         1         51          0         51     #    11
      160         9         51          0         51     #    19
      120        10         52          0         52     #    20
      120        11         52          0         52     #    21
      120        19         52          0         52     #    29
       80        20         53          0         53     #    30
       80        24         53          0         53     #    34
       80        25         54          0         54     #    35
       80        29         54          0         54     #    39
       40        30         55          0         55     #    40
       40        34         55          0         55     #    44
       40        35         56          0         56     #    45
       40        36         57          0         57     #    46
       40        37         58          0         58     #    47
       40        38         58          0         58     #    48
       40        40         58          0         59     #    50
       40        45         58          0         59     #    55
       40        46         58          0         59     #    56
       40        47         58          0         59     #    57
       40        48         58          0         59     #    58
       20        49         59      32000         59     #    59
```

SOLARIS laiko dalinimo klasės TS prioritetų kaita

- **ts_quantum**: *tai laikas pagal nutylėjimą skirtas tam tikram prioritetui.*
- **ts_tqexp**: *tai naujas prioritetas procesui , kuris pilnai išnaudojo laiko kvantą.*
- **ts_slpret**: *naujas prioritetas procesui, kuris užsiblokavo neišnaudojęs laiko kvanto.*
- **ts_maxwait**: *jei gija negauna CPU per laiką* ts_maxwait, *jos prioritetas paaukštinamas iki* ts_lwait.

Operacinės sistemos ir naudojamas planavimas

OS	Perimamas CPU (preemption)	Planavimo algoritmas
FreeBSD	Yes	Multilevel feedback queue
Linux pre-2.6	Yes	Multilevel feedback queue
Linux 2.6-2.6.23	Yes	O(1) scheduler
Linux post-2.6.23	Yes	Completely Fair Scheduler
Mac OS pre-9	None	Cooperative Scheduler
Mac OS 9	Some	Preemptive for MP tasks, Cooperative Scheduler for processes and threads
Mac OS X	Yes	Multilevel feedback queue
NetBSD	Yes	Multilevel feedback queue
Solaris	Yes	Multilevel feedback queue
Windows 95, 98, Me	Half	Preemptive for 32-bit processes, Cooperative Scheduler for 16-bit processes
Windows NT (including 2000, XP, Vista, 7, and Server)	Yes	Multilevel feedback queue

Algoritmų įvertinimas

- Technikos, naudojamos kompiuterinių sistemų įvertinimui:
- Determinuotas modeliavimas
  - Kiekvienas būvis vienareikšmiškai yra nusakomas parametrų reikšmėmis ir buvusiais būviais.
- Eilių modeliai
  - Matematiniai modeliai, kurių pagalba skaičiuojami laukiami sistemos parametrai
- Imitaciniai modeliai
  - Algoritminiai modeliai, kurie imituoja supaprastintą sistemos versiją, naudojant statistinius įėjimo duomenis.

Ačiū už dėmesį