Operacinės sistemos

N. Sarafinienė

2013m.

Kalbėsime

- CPU laiko planavimą
- Jam keliamus tikslus
- Taikomus algoritmus
- Planavimą realaus laiko sistemose bei multiprocesorinėse sistemose

Planuojama atsižvelgiant į procesų tipus

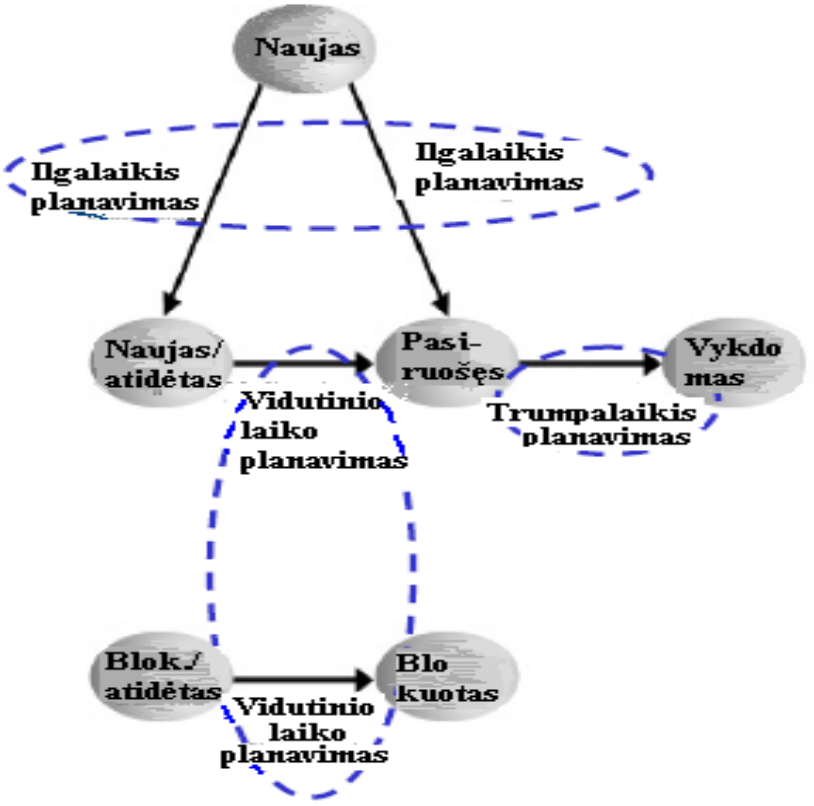
- Tradiciškai procesai skirstomi:
  - I/O bound – daug laiko sugaišta ties I/O įrenginiais bei laukdami šių įrenginių eilėse
  - CPU-bound – atlieka daug skaičiavimų
- Alternatyvi procesų klasifikacija:
  - Interaktyvūs procesai
  - Paketinio tipo
  - Realaus laiko

CPU laiko planavimui keliami uždaviniai:

- optimaliai išnaudoti CPU,
- garantuoti maksimalų CPU pralaidumą,
- užtikrinti tai, kad bendras CPU atliktų darbų kiekis bus maksimalus.

trijų tipų planavimas

- Ilgalaikis planavimas
- Vidutinio laiko planavimas
- Trumpalaikis planavimas.



Ilgalaikis planavimas

- Tai užduočių planavimas
- Sprendžia:
  - kada ir kokia tvarka priimti į sistemą naujus procesus
  - kuri programa bus priimama į sistemą ir bus įtraukta į eilę (sąrašą) „**Naujas**“.
- 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
- Sprendžia:
  - kada ir kokį procesą reikia ***atidėti*** (iškelti iš pagrindinės atmintinės)
  - kada ir kokį procesą reikia ***suaktyvinti*** (įkelti į pagrindinę atmintinę).
- Ilgalaikio ir vidutinio laiko planavimo mechanizmai kontroliuoja ***multiprogramavimo lygį***. t.y. apsprendžia vienu metu vykdomų procesų kiekį.

Trumpalaikis planavimas

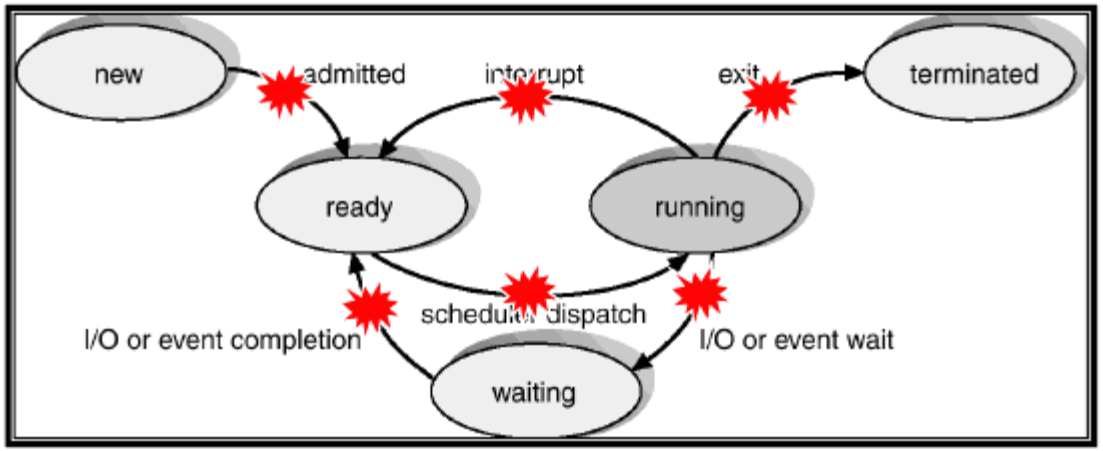
- ***Sprendžia, kurio proceso vykdymui bus skiriamas CPU- tai CPU planuotojas*.**
- Šis planavimas vykdomas įvykus vienam iš šių įvykių:
  - ***Laikrodžio*** mechanizmo generuota pertrauktis.
  - I/O ***pertrauktis***.
  - Sutiktas OS ***kvietinys***
  - ***Signalas***
- Šio planavimo rezultate iš pasiruošusių 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)

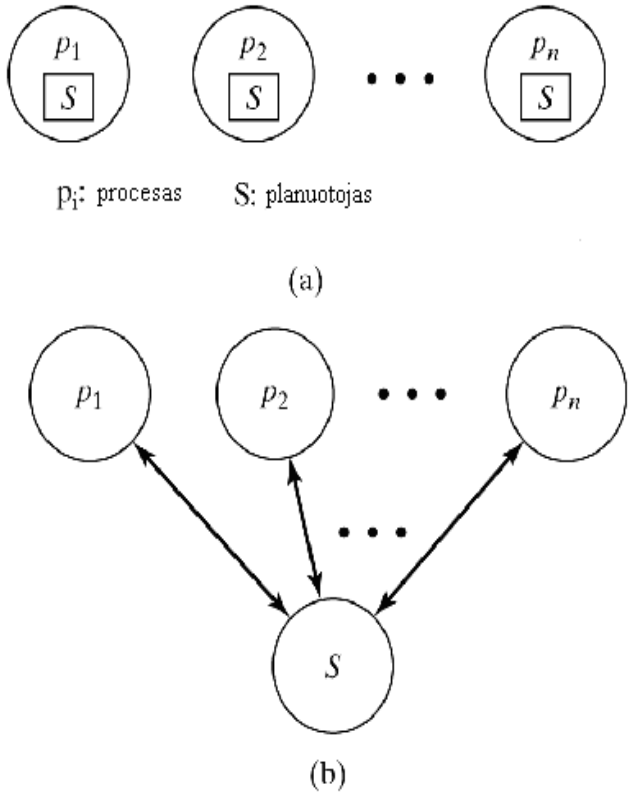
Planavimo taškai

- Planuotojas paprastai įsijungia, kai:
  - Užduotis perjungiama iš vykdymo būsenos į laukimo
  - Kai pasirodo pertraukimas
  - Kai yra sukuriama ar užbaigiama užduotis



Galima planuotojų realizacija

- planuotojai gali būti realizuoti kaip *įjungiamos* procedūros (a pav.) arba kaip *atskiri* procesai (b pav.)



CPU planavimas

- Multiprogramavimas ir planavimas
  - Multiprograminėje sistemoje stengiamasi padidinti CPU panaudojimą ir užduočių pralaidumą *perdengiant I/O bei CPU aktyvumus*
  - Mechanizmai
    - ***Konteksto*** *perjungimas* (kada ir kaip tai vykdoma)
    - ***Procesų eilės*** ir procesų būviai
  - Politika
    - Kurį procesą (giją) vykdyti, kiek ilgai, t.t.

CPU planuotojas

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

CPU Perėmimas

- Kitas su planavimu susijęs momentas – tai CPU perėmimo galimybė. Yra dvi galimybės:
  - **Negalimas perėmimas**
    - Planuotojas laukia kol vykdoma užduotis ***užsiblokuos***.
    - Planavimas vykdomas tik tada, kai:
      - Kai procesas persijungia iš vykdymo būsenos į ***laukimo būseną***
      - Kai procesas ***pasibaigia***
  - **Galimas perėmimas**
    - Planuotojas gali pertraukti užduotį ir iššaukti konteksto perjungimą.
    - Kas atsitinka, jei:
      - Perimamas procesas kuris keitė *bendrai naudojamus duomenis*?
      - Jei procesas *vykdė sisteminį kvietinį*?
    - Tai geriau tinkama politika, nes neleidžia jokiam procesui monopolizuoti procesorių ilgame laiko intervale.

Planavimo tikslai (1)

- Visose sistemose
  - **Sąžiningumas** (fairness) – kiekvienam procesui suteikti galimybę gauti CPU laiko
  - **Balansas** – visus sistemos įrenginius turėti apkrautus
- *Paketinio tipo užduočių apdorojimas*
  - **Pralaidumas**: max užduočių kiekis per valandą
  - **Pilnas įvykdymo** (Turnaround) **laikas:** min laiką nuo užduoties pateikimo iki įvykdymo.
  - **CPU panaudojimas**: palaikyti CPU apkrautą visą laiką

Planavimo tikslai (2)

- *Interaktyviose sistemose*
  - **Atsakymo laikas**
  - **Laukimo laikas**
  - Atitikti vartotojų siekiamybes
- *Realaus laiko sistemos*
  - **Kritinių laiko ribų išlaikymas**
  - **Nusakomumas** (predictability)

Planuojant vengiama

- Badavimo
  - Tai situacija kai procesas nedaro jokio progreso kadangi kitas procesas yra užėmęs resursą, kurio jam reikia
    - Resursu gali būti CPU, atmintinė, t.t.
  - Bloga planavimo politika gali iššaukti badavimą
    - Jei aukšto prioriteto procesas visad užgrobs CPU neleisdamas žemo prioriteto procesui jį gauti
  - Badavimą gali sukelti **sinchronizavimas**
    - Vis atsiranda skaitančių procesų ir jie blokuoja rašančius.

Planavimo tikslai

- Jie priklauso nuo konkrečios operacinės sistemos ir gali būti tokie:
  - Maksimalus sistemos pralaidumas. (Kompiliatoriai, skaičiavimo tipo procesai)
  - Maksimalus kiekis interaktyvių procesų, kuriems garantuojamas priimtinas atsakymo laikas (Minimalūs vėlinimo laikai).
  - Procesų įvykdymas iki nustatyto laiko momento (Realaus laiko sistemos)
  - Efektyvus sistemos išteklių panaudojimas.
  - Proceso vykdymo atidėjimo neapibrėžtam laikui išvengimas.
  - Prioritetinis aptarnavimas.
  - Minimalūs su planavimu susiję veiksmai.

CPU planuotojo kriterijai

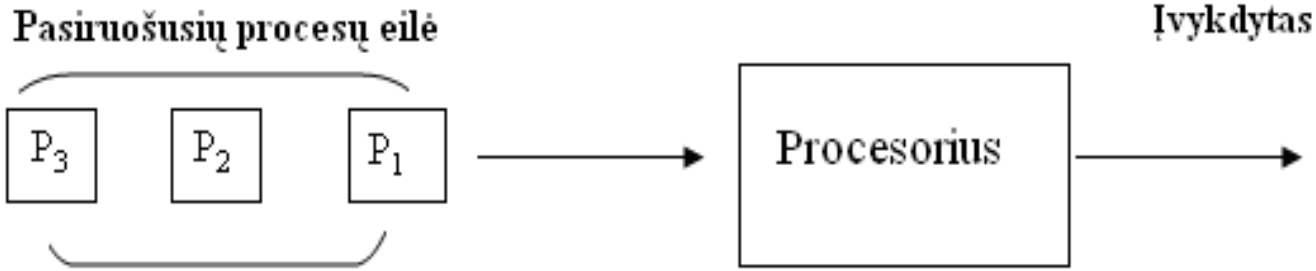
- Pagrindiniai taikomi kriterijai yra šie:
  - **CPU panaudojimas**, kuris yra nusakomas procentais išreikšta laiko dalimi, kurios metu procesorius yra užimtas.
  - **Pralaidumas.** Jis yra nusakomas kiekiu procesų, kurie baigia savo vykdymą per nurodytą laiko vienetą.
  - **Atsakymo laikas** – laiko tarpas nuo to momento, kai užklausa buvo pateikta iki tol, kol buvo gauti pirmi atsakymai.
  - **Pilnas įvykdymo** (Turnaround) **laikas** – laikas, kurio reikia įvykdyti tam tikrą procesą .

Planavimo algoritmai

- **Pirmas** pasirodęs aptarnaujamas pirmas (FCFS)
- **Ciklinis** - Round-Robin (RR)
- **Trumpiausias** procesas aptarnaujamas pirmas - (SPF)
- Procesas su **trumpiausiu likusiu** aptarnavimo laiku – pirmas (SRT)
- **HRRN** disciplina
- **Prioritetinis** aptarnavimas

FCFS disciplina

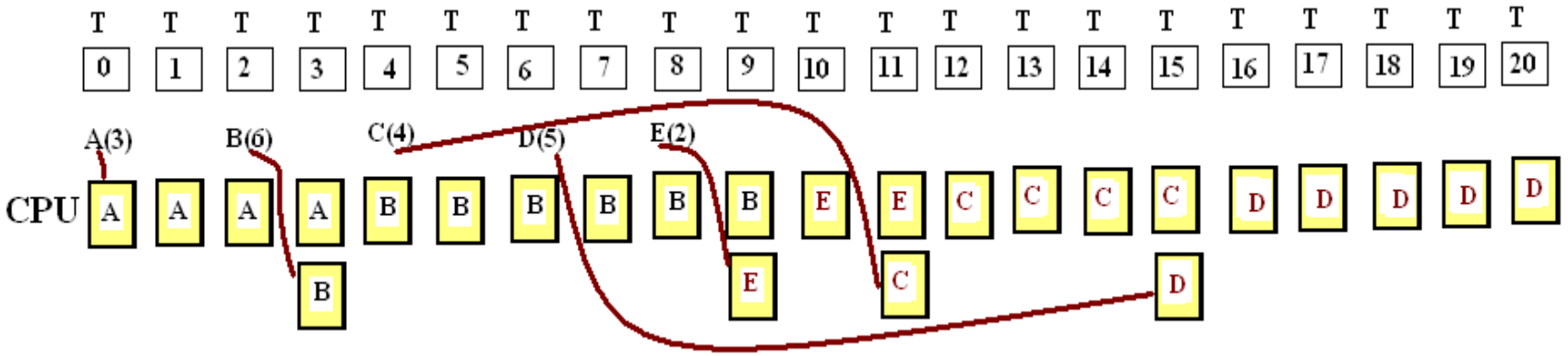
- FCFS- pirmas pasirodęs procesas yra ir aptarnaujamas pirmas.
- Naudojant FCFS discipliną ***perėmimas nėra galimas***.
- Gerai tinka daugiau **skaičiavimų** pobūdžio (CPU-bound) tipo procesams, ***nėra paranki*** trumpiems procesams.
- Užduotys traktuojamos vienodai – **badavimo nėra**
- Problemos:
  - Vidutinis laukimo laikas gali būti didelis, jei trumpos užduotys laukia už ilgų
  - Gali gautis blogas persidengimas tarp I/O ir CPU veiksmų



Trumpiausias procesas pirmas

- Trumpiausias procesas aptarnaujamas pirmas (SPN-SJF): aptarnavimui yra renkamas procesas, kurio laukiamas įvykdymo laikas yra mažiausias.
  - Tai leidžia sumažinti laukiančių procesų kiekį.
  - Pats algoritmas yra ***nepertraukiamo*** tipo.
  - Jis reikalauja įvertinti, kiek laiko reikės proceso įvykdymui
- Taikant šį algoritmą gaunami *mažesni vidutiniai laukimo* laikai.
- Algoritmas gerai tinka ***trumpų*** procesų aptarnavimui.
- Netinka modernioms interaktyvioms sistemoms.
- **Problemos**:
  - Sunku įvertinti CPU poreikio dydį, gali sukelti badavimus

Trumpiausias procesas aptarnaujamas pirmas



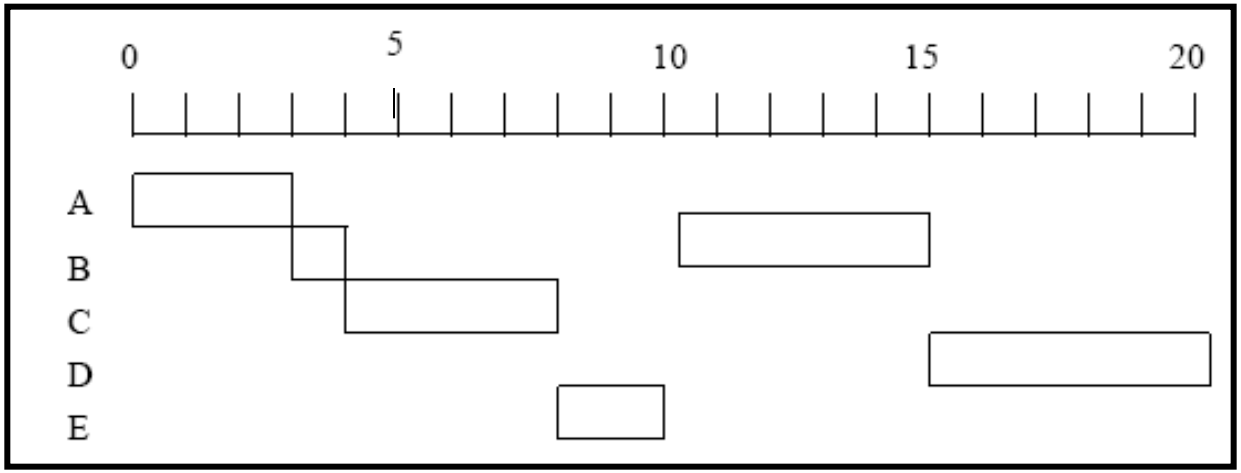
Procesas	Pasirodymo laikas	Aptarnavimo laikas
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

CPU nėra perimamas.
•Trumpi procesai iššoka prieš ilgesnius.

Likęs mažiausias aptarnavimo laikas aptarnaujamas pirmas SRT

- Aptarnavimui parenka procesą, įvertinant likusį aptarnavimo laiką.
- Šis algoritmas yra SPN versija, tačiau ji jau teikia ***CPU perėmimo*** galimybę:
  - Jei atėjęs naujas procesas turi mažesnį CPU pareikalavimą, procesas yra pertraukiamas
- SRT algoritmas pasižymi šiomis charakteristikomis:
  - Duoda ***aukštą pralaidumą***.
  - Užtikrina ***minimalų*** (optimalų) ***vidutinį atsakymo*** (laukimo) laiką duotam procesų rinkiniui.
- Reikalinga įvertinti likusį reikalaujamą CPU laiką :
  - Atliekama automatiškai (naudojant eksponentinį vidurkių skaičiavimą).
  - Vertinamas vartotojo pateiktas procesui reikalingas laikas.
- SRT algoritmo atveju ***ilgesni procesai gali "badauti"*** nesulaukdami CPU. Atsiranda problema ilgi procesai gali būti niekad nevykdomi.

Likęs mažiausias aptarnavimo laikas aptarnaujamas pirmas

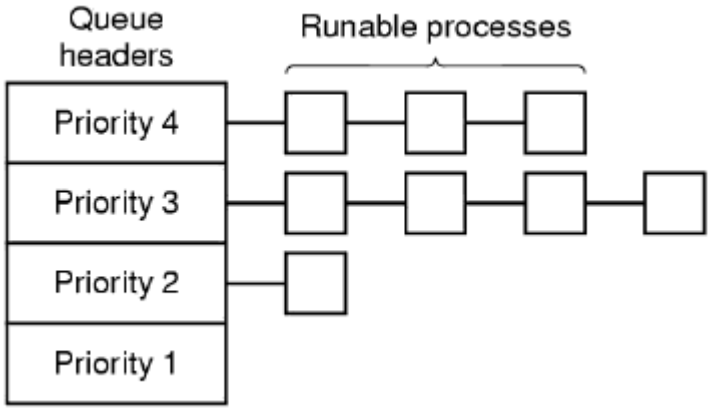


Procesas	Pasirodymo laikas	Aptarnavimo laikas
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

CPU perimamas (proceso pasirodymo momentu), jei sekantis procesas yra trumpesnis.

Prioritetinis planavimas (1)

- Prioritetai įvertina procesų svarbą.
- Atsiradus pasiruošusių procesų eilėje procesui su didesniu prioritetu :
  - Proceso aptarnavimas gali būti ***nutraukiamas*** ir pradedamas vykdyti aukštesnio prioriteto procesas
  - Vykdomam procesui gali būti ***leidžiama užbaigti*** išnaudoti jam skirtą procesoriaus laiko kvantą ir tik po to pereinama prie proceso su aukštesniu prioritetu vykdymo.
- Prioritetas gali būti nusakomas ir eile, į kurią procesai surikiuojami
- Prioritetai gali būti **statiniai** arba **dinaminiai**.



Prioritetinis planavimas (2)

- Badavimo problema
  - Jei visą laiką bus aukšto prioriteto procesų, žemo prioriteto procesai badaus
- Sprendimas: sendinimas
  - Prioritetas *didinamas* augant laukimo laikui
  - Prioritetas *mažinamas* priklausomai nuo gauto CPU laiko
  - Taikomi kiti euristiniai algoritmai

Prioritetinis planavimas-(2-1)

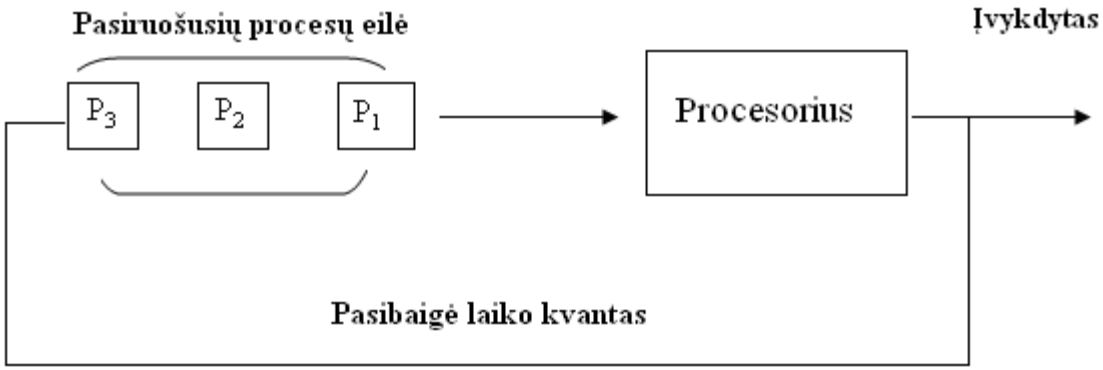
- Nusakant proceso prioritetą gali būti vertinami šie dydžiai:
  - ***Gautas*** aptarnavimo laikas.
  - Realus proceso ***gyvavimo*** sistemoje laikas.
  - ***Bendras*** reikalaujamas proceso aptarnavimo laikas.
  - ***Vartotojo*** nusakomas proceso prioritetas (*nice*).
  - ***Periodas***, kurio metu procesas turi būti vykdomas.
  - Laiko ***momentas***, iki kurio procesas turi būti baigtas.
  - Proceso pareikalavimai ***atmintinės*** dydžiui.

Prioritetinis planavimas (3)

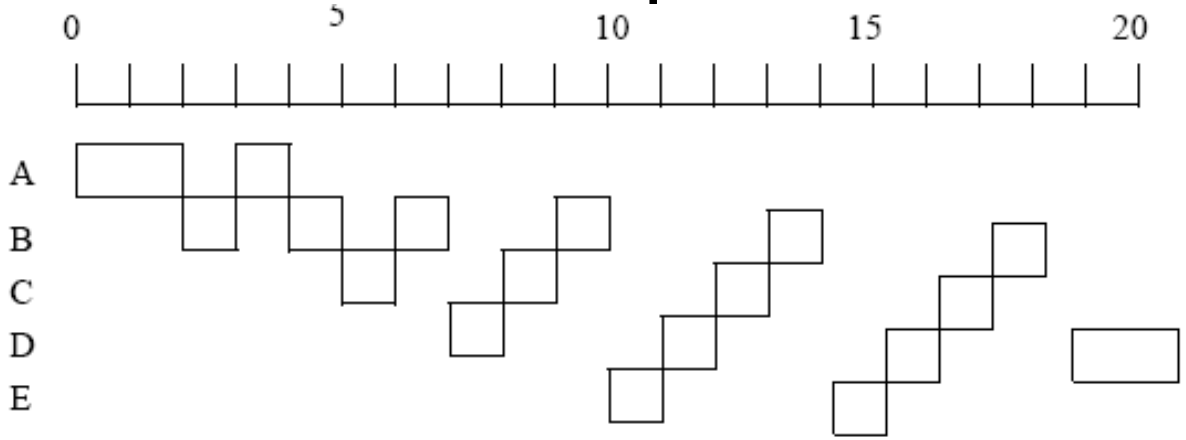
- Prioriteto inversijos problema
  - Tai situacija, kai aukštesnio prioriteto užduotis negali būt vykdoma, nes žemesnio prioriteto užduotis turi užėmusi resursą, kurio reikia aukštesnio prioriteto užduočiai (pvz. mutex)
- Sprendimas: prioriteto paveldimumas
  - Aukštesnio prioriteto užduotis gali perduoti savo prioritetą žemesnio prioriteto užduočiai, turinčiai reikiamą resursą, ji turės aukštesnį prioritetą ir bus greičiau įvykdyta, o ją įvykdžiusi ji grąžins prioritetą užduočiai, iš kurios jį buvo perėmusi.

Ciklinis aptarnavimas (Round Robin algoritmas)

- Siekiant sumažinti procesoriaus laukimo laiką trumpiems procesams yra naudojama laiko dalinimo schema.
  - Planuotojas priskiria ***laiko kvantus*** (q) kiekvienam iš procesų,
  - jei šio laiko kvanto nepakanka, tai procesas turi laukti, kol jam bus išskirtas sekantis laiko kvantas.
  - q- paprastai 10-100 msec
- Šio algoritmo atveju pagrindine problema yra ***tinkamo laiko kvanto dydžio parinkimas***
- RR disciplina yra ***efektyvesnė*** nei FCFS, ***gaunamas*** didesnis vidutinis pilno įvykdymo laikas nei SJF, bet ***geresni atsakymo*** laikai; ***nėra badavimo***



Round Robin –ciklinis aptarnavimas



Procesas	Pasirodymo laikas	Aptarnavimo laikas
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

CPU perėmimas pagal laikrodžio mechanizmą (pertraukimas vyksta kas laiko kvantą -q-)

• Esant n procesų, kiekvienas iš jų gauna 1/n dalį CPU laiko kvantais, kurio kiekvieno maksimalus dydis q.

• Funkcionavimas

– q didelis → FIFO

– q mažas →gaištama daug laiko perėjimams nuo vieno proceso prie kito.

HRRN disciplina

- HRRN (Highest Response Ratio Next) –atrenka vykdymui procesą, turintį didžiausią atsakymo laiko proporciją.
- Ši disciplina įvertina tai, kiek laiko procesas jau yra sistemoje ir siekia išvengti badaujančių procesų atsiradimo.
- Kiekvienam procesui yra paskaičiuojamas santykis:

$$R = \frac{w+s}{s}$$

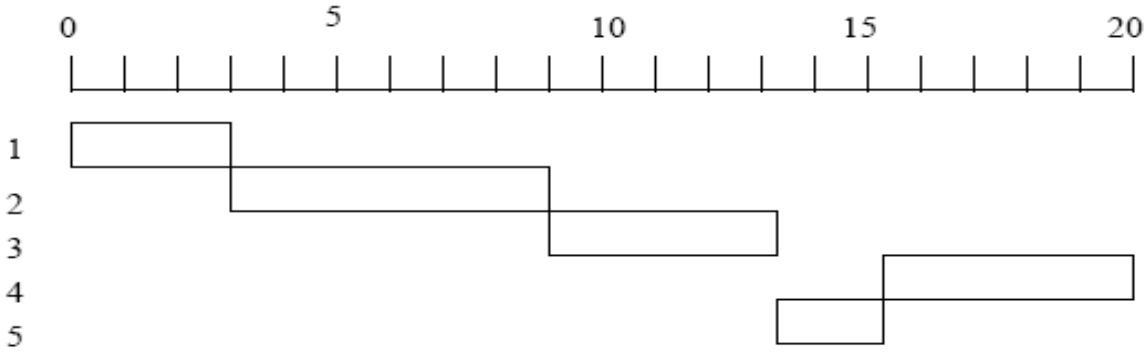
w – yra laikas, praėjęs nuo proceso sukūrimo,

s- laukiamas aptarnavimo laikas.

Sekantis parenkamas vykdymui procesas bus tas, kuriam šis santykis bus didžiausias.

Šio algoritmo atveju CPU **nėra perimamas**.

Prioritetinis planavimas
HRRN (Highest Response Ratio Next) disciplina



$$\frac{\text{laikas praleistas laukiant + laukiamas aptarnavimo laikas}}{\text{laukiamas aptarnavimo laikas}}$$

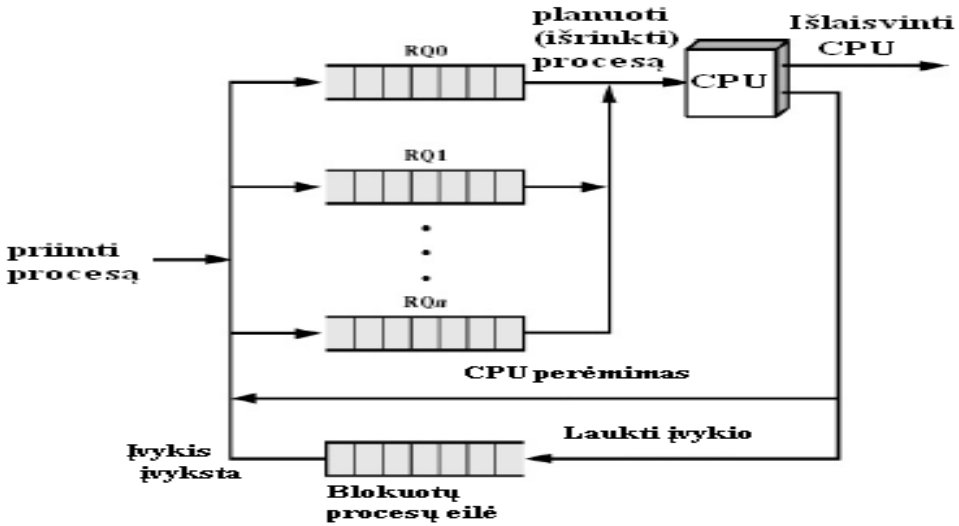
Procesas	Pasirodymo laikas	**Aptarnavimo laikas**
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Aptarnavimui parenkamas procesas, kuriam šio santykio reikšmė yra didžiausia.

- CPU **nėra** perimamas
- **Nėra** badavimo (įvertinamas sendinimas)
- **Tinka** trumpiems procesams.
- Gali iššaukti **daug** papildomų veiksmų.

Daugelio lygių eilė (1)

- Pasiruošusių procesų eilė gali būti sudaloma į atskiras eiles, į jas talpinami skirtingo tipo procesai
  - Kiekviena eilė gali turėti ***savą procesų planavimo discipliną***,.
  - Taikomas tam tikras mechanizmas ***planavimui*** tarp eilių



Prioritetinės eilės

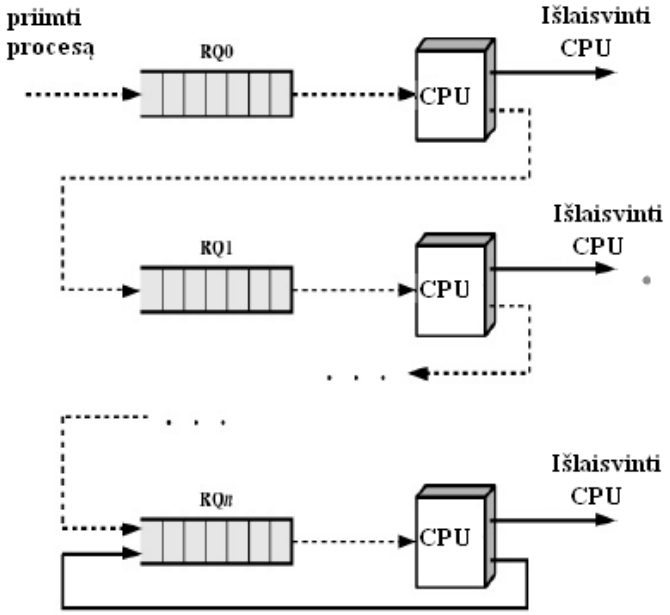
Daugelio lygių eilė (2)

- Planavimas tarp eilių
  - Jis gali būti **fiksuotas**, pavyzdžiui, aptarnauti pradžioje visus interaktyvius procesus, tada foninius.
    - galimas badavimas.
  - 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.

Daugelio lygių grįžtamojo ryšio eilė-(aging algoritmas)

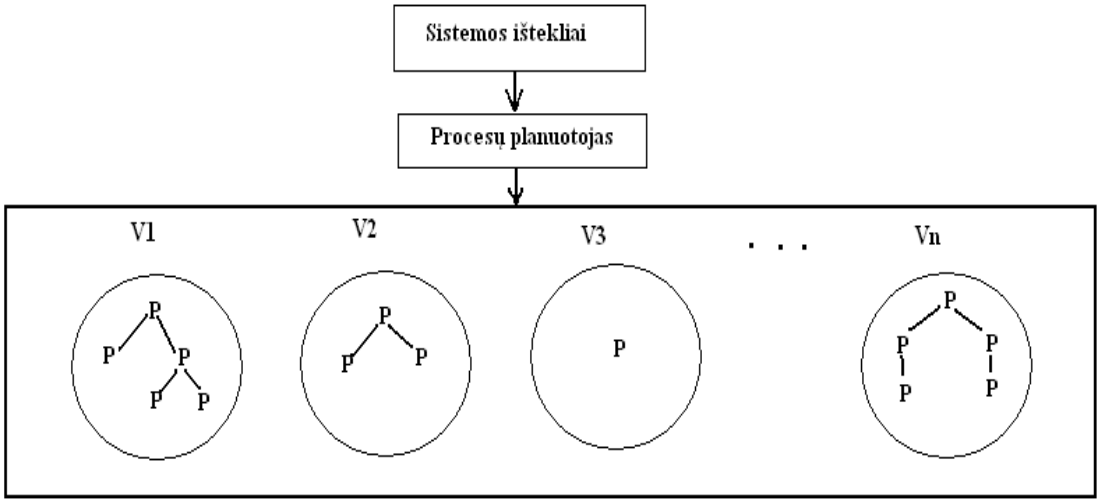
- Procesas gali būti nepririšamas prie kažkurios konkrečios eilės, o pereiti per įvairias eiles, kurioms gali būti taikomas skirtingas planavimas.

- **planuotojo parametrais** gali būti:
  - **Eilių kiekis**.
  - Planavimo algoritmas, taikomas kiekvienai eilei.
  - Proceso **naujinimo** metodas (per ilgai laukiant).
  - Proceso „**žeminimo**“ metodas (gavus CPU laiko).
  - Metodas, kuris apsprendžia į kurią eilę **pradžioje** turi patekti procesas.



Fair-Share (sąžiningo CPU dalinimosi) planavimas

- Tai išteklių dalinimosi tarp grupių disciplina.
- Planavimo sprendimai grindžiami atžvilgiu procesų grupės, o ne atskiro proceso atžvilgiu.
- Ši disciplina kontroliuoja vartotojų prieigą prie sistemos išteklių.
- Tai standartinis planavimas, kuris taikomas UNIX operacinėse sistemose, procesoriaus laikas yra dalomas tarp vartotojų, nesvarbu kiek procesų jie turėtų



UNIX planuotojas (1)

- Unix planuotojas naudoja **daugelio lygių grįžtamojo ryšio eiles**
  - 3-4 užduočių klasės:
    - laiko dalinimo, sisteminės, realaus laiko, pertraukimo
    - apie 170 prioritetų
  - Prioritetinis planavimas tarp eilių,
  - RR eilės viduje

  - Procesams prioritetai keičiami dinamiškai:
    - Didėja, jei procesas užblokuojamas neišnaudojęs kvanto
    - Mažėja, jei išnaudoja kvantą pilnai

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.
-
- Solaris 9 įdiegtos FSS ir FX kasės.

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

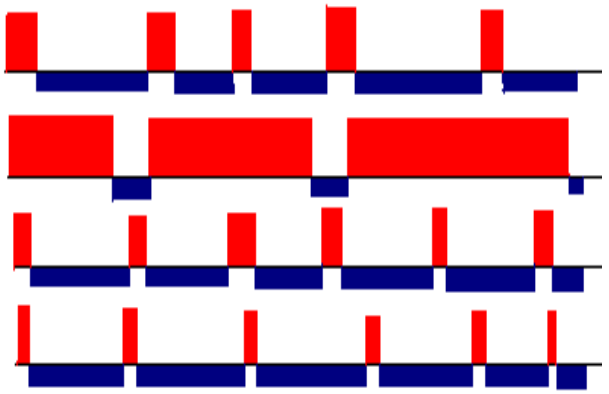
# ts_quantum	ts_tqexp	ts_slpret	ts_maxwait	ts_lwait	PR-	LEVEL
200	**0**	**50**	**0**	**50**	**#**	**0**
200	0	50	0	50	#	1
200	0	50	0	50	#	2
200	0	50	0	50	#	9
160	**0**	**51**	**0**	**51**	**#**	**10**
160	1	51	0	51	#	11
160	9	51	0	51	#	19
120	**10**	**52**	**0**	**52**	**#**	**20**
120	11	52	0	52	#	21
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.

UNIX planuotojas (2)

- Motyvacija
  - Interaktyvūs procesai paprastai vyksta naudodami trumpus CPU laiko pliūpsnius.
    - Jie pilnai neišnaudoja kvanto prieš pradėdami laukti I/O
    - Idėja- išstumti interaktyvius procesus prieš tuos, kurie naudoja daug CPU laiko
  - Norima minimizuoti atsakymo laikus
    - Tai laikai kai baigiama įvest iš klaviatūros iki procesas pradedamas vykdyt
    - Nesinori, kad redaguojant reiktų laukt, kol ilgas skaičiavimo tipo procesas baigs darbą
  - Tokia politika duoda ***vėlinimus skaičiuojamojo pobūdžio*** procesams



Linux planuotojas (1)

- Bendros charakteristikos
  - Linux siūlo tris planavimo algoritmus
    - Tradicinį UNIX planuotoją: SCHED_OTHER
    - Du realaus laiko planuotojus (POSIX.1b):SCHED_FIFO ir SCHED_RR
  - Linux planavimo algoritmai realaus laiko procesams yra “soft real time”
    - Jie teikia pirmenybę realaus laiko procesams, jei yra realaus laiko procesų
    - Priešingu atveju jie perduoda CPU kitiems procesams

Linux planuotojas (2)

- Prioritetai
  - Statiniai
    - Nusako max kvanto dydį, skiriamą procesui, po kurio yra leidžiama kitiems procesams varžytis dėl CPU
  - Dinaminiai
    - Kas kažkiek laiko prioritetas perskaičiuojamas
      - Jei negauna CPU, prioritetas ***didinamas***, gavusiems – ***mažinamas***
  - Realaus laiko prioritetai
    - Tik realaus laiko procesams suteikiamas realaus laiko prioritetas
    - Aukštesnio realaus laiko prioritetas “skriaudžia” žemesnį

Linux planuotojas (3)

užduoties struktūros laukai

`long counter;`	Likęs einamasis kvanto dydis (nusako dinaminį prioritetą)
`long nice;`	Užduoties "nice" reikšmė, nuo -20 iki +19. Nusako statinį prioritetą
`unsigned long policy;`	SCHED_OTHER, SCHED_FIFO, SCHED_RR
`struct mm_struct * mm;`	Nuoroda į atmintinės deskriptorių
`int processor;`	Procesoriaus ID, kuriame užduotis bus vykdoma
`unsigned long cpus_runnable;`	~ 0, jei užduotis nevykdoma nė jokiame CPU (1 << cpu) jei vykdoma CPU
`unsigned long cpus_allowed;`	CPU, kuriuose gali būt vykdoma
`struct list_head run_list;`	pasiruošusių procesų eilės pradžia
`unsigned long rt_priority;`	realaus laiko prioritetas

Linux planuotojas (4)

- Planavimo politikos
  - SCHED_OTHER
  - SCHED_FIFO
    - Realaus laiko procesai vykdomi tol, kol jie arba ***užsiblokuoja*** laukdami I/O, arba ***grąžina*** CPU (yield) arba yra ***pertraukiami*** aukštesnio prioriteto proceso.
    - Veikia tarsi nebūtų laiko kvanto
  - SCHED_RR
    - Tai tas pats kaip ir SCHED_FIFO, išskyrus tai, kad vyksta laiko dalinimas
    - Kai SCHED_RR proceso laiko kvantas baigiasi, jis grįžta į sąrašą procesų su tuo pačiu prioritetu.

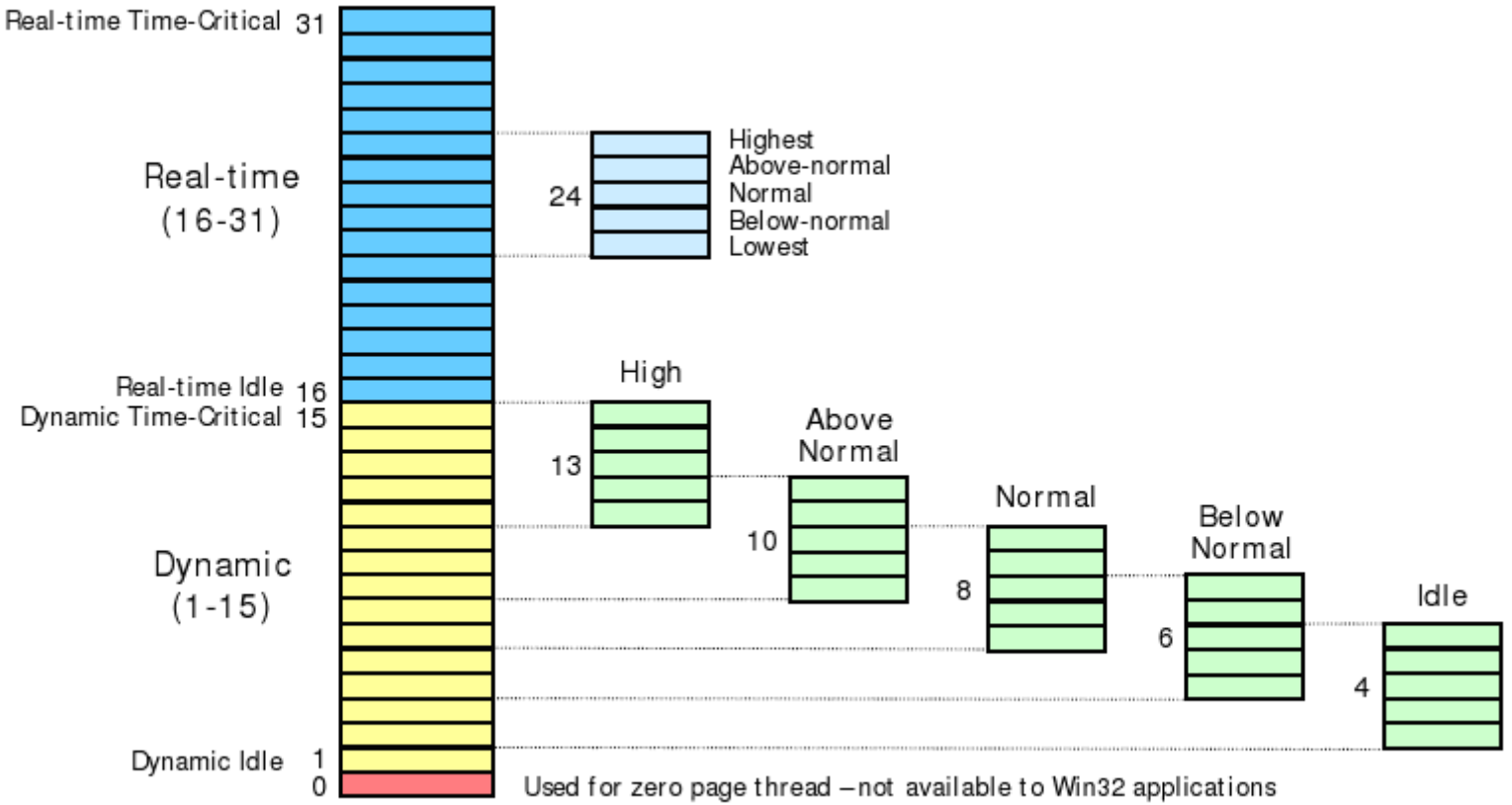
Linux planuotojas (5)

- Epochos
- Linux planuotojas veikia dalindamas ***CPU laiką į epochas***.
  - Vienoje epochoje kiekvienas procesas turi nusakytą laiko kvantą, kurio ilgis paskaičiuojamas prasidėjus epochai.
  - Epocha baigiasi kada visi pasiruošę vykdymui procesai yra išnaudoję savo kvantą.
  - Planuotojas perskaičiuoja laiko kvantų ilgius visiems procesams ir prasideda nauja epocha.
- Bazinis laiko kvanto dydis procesui yra paskaičiuojamas remiantis “nice” reikšme.

Windows 2000 planuotojas (1)

- Planavimas Windows 2000 sistemoj
  - Planavimas vyksta gijų lygyje
  - Jis paremtas prioritetais, pertraukimas galimas
    - Aukštesnio prioriteto gija visada yra vykdoma.
    - Laiko dalinimas, round-robin prioriteto eilėj.
  - Windows 2000 naudoja 32 prioriteto lygius
    - Sisteminis lygis (0), kintami lygiai (1-15), realaus laiko (16-31)
  - Iš Win32 požiūrio taško
    - Procesams suteikiama prioriteto klasė jį sukūrus:
      - Tuščias (idle), žemiau normalaus, normalus, virš normalaus, aukštas, realaus laiko
      - Keičiamas užduočių valdytojo
    - Individuali gija turi reliatyvų prioritetą klasės viduje
      - Tuščias (idle), žemiau normalaus, normalus, virš normalaus, aukščiausias, laiko atžvilgiu kritinis

Windows 2000 planuotojas (2)



Windows 2000 planuotojas (3)

- Windows 2000 – laiko kvantas
  - Pagal nutylėjimą gijos startuoja su kvanto reikšme:
    - 6 Windows 2000 Profesional
    - 36 Windows 2000 server
  - Windows 2000 server atveju ilgesnis kvanto dydis suteikiamas siekiant minimizuoti konteksto perjungimą
  - Laikrodžio pertraukimai daugumoje x86 procesorių (vieno branduolio) yra kas 10ms, ir daugumos x86 multiprocesorių -15ms.
  - Kiekvieną kartą esant pertraukimui pagal laikrodį, laikrodžio pertraukimus apdorojanti funkcija atima fiksuotą dydį (3) iš gijos laiko kvanto

Algoritmų palyginimas

Scheduling algorithm	CPU Utilization	Throughput	Turnaround time	Response time	Deadline handling	Starvation free
First In First Out	Low	Low	High	High	No	Yes
Shortest remaining time	Medium	High	Medium	Medium	No	No
Fixed priority pre-emptive scheduling	Medium	Low	High	High	Yes	No
Round-robin scheduling	High	Medium	Medium	Low	No	Yes

Operacinės sistemos ir naudojamas planavimas

Operating System	Preemption	Algorithm
Windows 3.1x	None	Cooperative Scheduler
Windows 95,98,ME	Half	Only for 32 bit operations
Windows NT,XP,Vista	Yes	Multilevel Feedback Queue
Mac OS pre 9	None	Cooperative Scheduler
Mac OS X	Yes	Mach (kernel)
Linux pre 2.5	Yes	Multilevel Feedback Queue
Linux 2.5-2.6.23	Yes	O(1) scheduler
Linux post 2.6.23	Yes	Completely Fair Scheduler
Solaris	Yes	Multilevel feedback queue
NetBSD	Yes	Multilevel feedback queue
FreeBSD	Yes	Multilevel feedback queue

O(1) algoritmas

- Yra išrenkama aukščiausio prioriteto eilė turinti bent vieną procesą:
  - Tai trunka tarkim t1 laiko vienetą
  - Pirmas procesas, esantis šio prioriteto sąraše (doubly-linked) yra parenkamas vykdymui – tai trunka tarkim t2 laiko vienetus
  - Bendras proceso išrinkimo laikas yra T= t1+t2
- Naujo proceso parinkimo vykdymui skirtas laikas yra fiksuotas – jis nepriklauso nuo procesų kiekio

Algoritmų įvertinimas

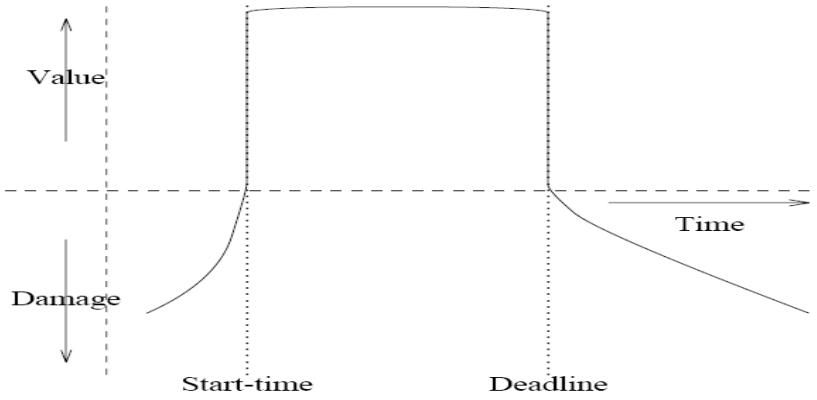
- Technikos, naudojamos kompiuterinių sistemų įvertinimui:
- Determinuotas modeliavimas
  - Kiekvienas būvis vienareikšmiškai yra nusakomas parametrų reikšmėmis ir buvusiais būviais.
- Eilių modeliai
  - Matematiniai modeliai, kurių pagalba skaičiuojami laukiami sistemos parametrai
- Imitaciniai modeliai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

Realaus laiko sistemos

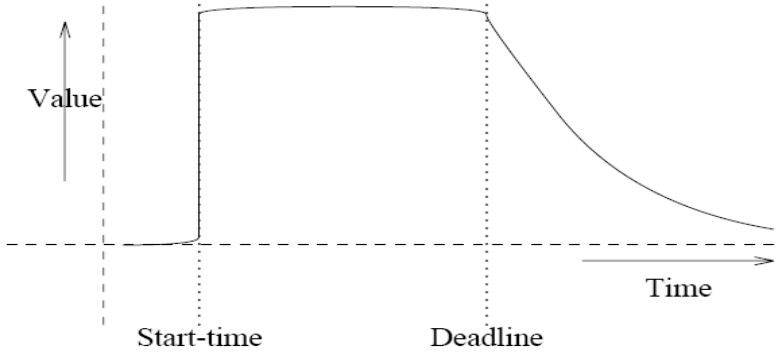
- Skaičiavimai vyksta reaguojant į išorės įvykius. Jie gali būti:
  - *periodiniai* (pav. koks nors pasikartojantis laike veiksmas)
  - *aperiodiniai* (pav. mygtuko paspaudimas)
- Realaus laiko sistemos skirstomos :
  - ***Kietų ribų*** (Hard) realaus laiko sistemos
    - – žinoma kritinė riba (deadline), kurią peržengus atsakymas tampa beverčiu
    - – veiksmų atlikimas gali būt nusakomas užduodant minimalius ar maksimalius laikus
  - ***Minkštų ribų*** (Soft) realaus laiko sistemos
    - – Kritinė veiksmų įvykdymo riba (deadline) nusakoma apytiksliai
    - – Atsakymo reikšmingumas mažėja didėjant poslinkiui nuo kritinės ribos.

kietų ribų, minkštų ribų, bei hibridinių sistemų reikalavimai

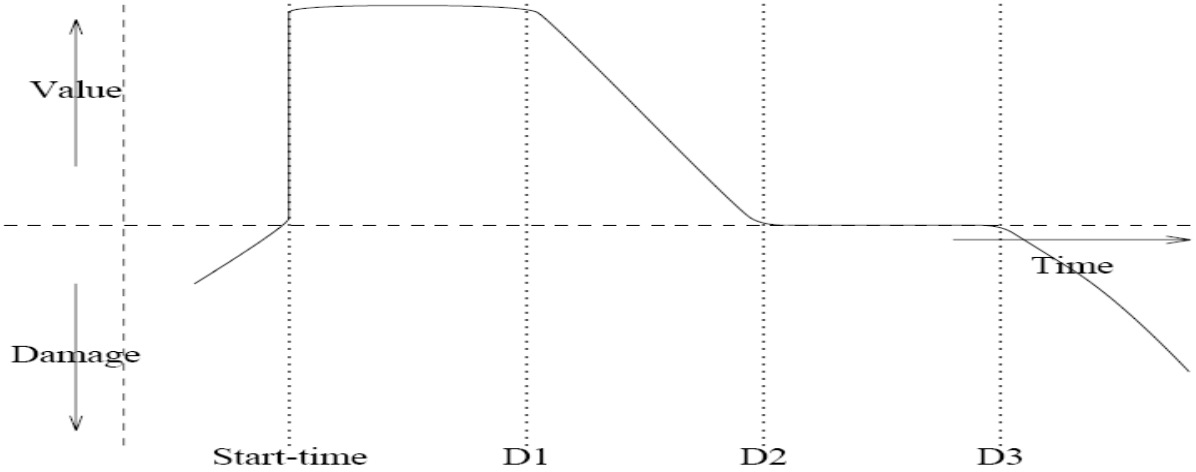
Kietos ribos



Minkštos ribos

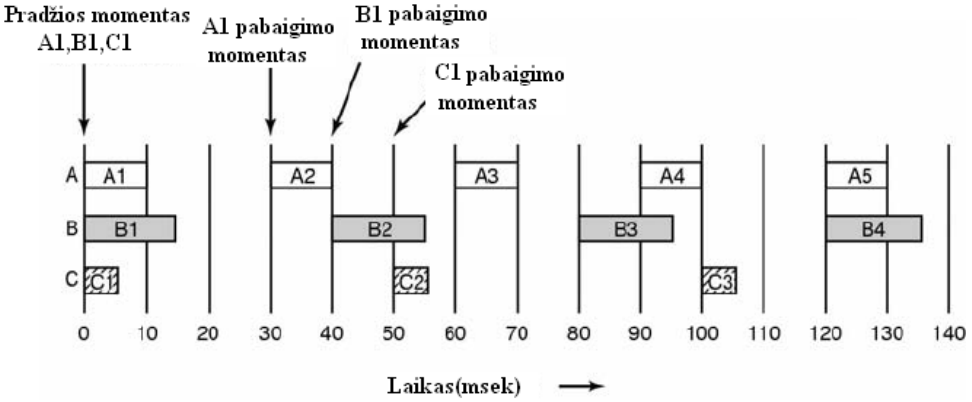


Hibridinės

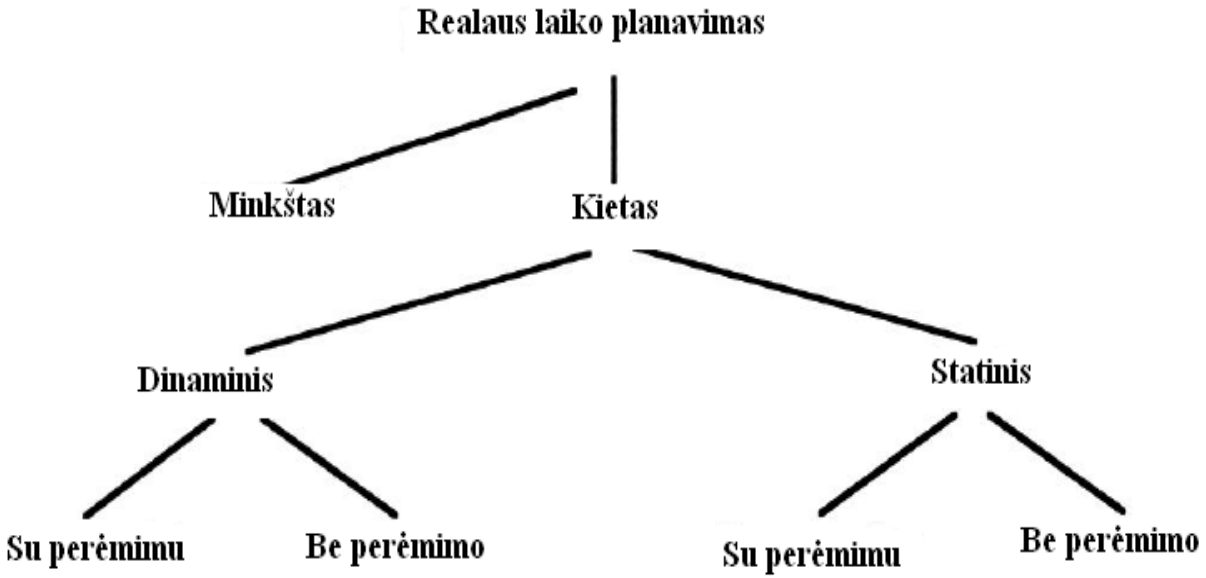


Realaus laiko sistemų apžvalga

- **Realaus laiko tai nereiškia realiai greitos**
  - Realaus laiko sistemos tai tokios, kurių rezultatų tikslumas priklauso nuo laiko kada yra gaunami rezultatai bei nuo jų funkcionavimo korektiškumo.
- ***Be minkštų ribų***, bei ***kietų ribų*** realaus laiko sistemų, kurios sužlunga, jei praleidžiamas kritinis laiko intervalas, dar egzistuoja ir tokios **(Firm real time)**, kuriose rezultatas už kritinio laiko intervalo neturi reikšmės, tačiau jos gali susitaikyti su tuo, kad yra praleisti kai kurie rezultatai.



Planavimas realaus laiko sistemose



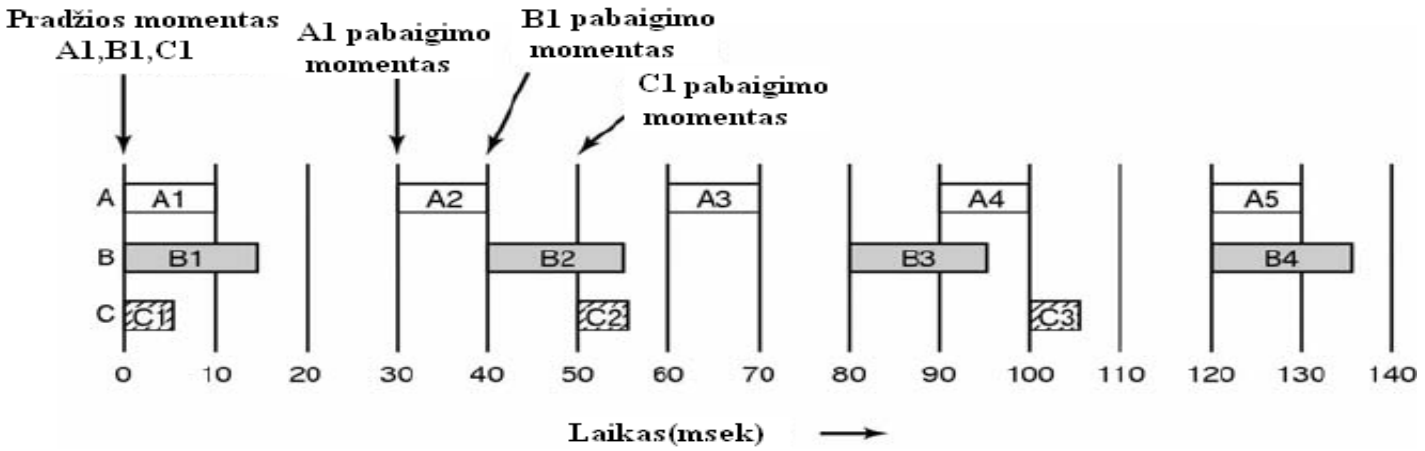
Realaus laiko sistemų planavimo tipai

- **Dinaminis - Statinis**
  - **Statinio** planavimo atveju, procesų prioritetai nėra reguliuojami laikui bėgant, todėl nėra papildomų veiksmų, susijusių su prioritetų perskaičiavimu
    Pav.: RMS (Rate Monotonic Scheduling) planavimo atveju procesams priskiriami prioritetai pagal jų kartojimosi periodo dydį, kuo jis trumpesnis, tuo didesnis prioritetas.
  - **Dinaminio** planavimo atveju, prioritetai yra perskaičiuojami reaguojant į besikeičiančias aplinkybes.
    - Pav.: didžiausias prioritetas gali būti suteikiamas procesui, kuris turi pasibaigti anksčiausiai.
- **Procesoriaus perėmimas** leidžia perimti CPU iš žemesnio prioriteto proceso.

Planavimo parametrai

- **Egzistuoja periodinių užduočių rinkinys {Ti}**
  - Periodai **pi**
  - Kritinės ribos (Deadline) **di**
  - Vykdymo laikai **ci** (CPU laiko kiekis, reikalingas užduoties įvykdymui)
- **Papildomi parametrai:**
  - Neapibrėžtumas (Laxity) **li = di – ci** (laisvo - slack laiko kiekis, prieš pradedant Ti *užduotes vykdymą*
  - Panaudojimo faktorius **ui = ci/pi** (nusako CPU panaudojimą)

- Daroma užduočių "**planuojamumo**" analizė, t.y. tiriama, ar esant šiam užduočių rinkiniui galima garantuoti kad galima bus užtikrinti jų įvykdymą iki kritinių ribų.



Planavimo daromos prielaidos

- **turime 5 apribojimus:**
  1. Užduotys {Ti} yra periodinės, su kietomis kritinėmis ribomis
  2. Užduotys yra pilnai nepriklausomos
  3. Kritinė riba = periodas pi = di
  4. Vykdymo laikas ci yra pastovus ir žinomas
  5. Konteksto persijungimas nevertinamas, nevertinant ir konteksto persiuntimo kitam CPU

Realaus laiko sistemų planavimas esant perėmimui

- Clock-Driven,
- Processor-Sharing,
- Priority-Driven.

Laikrodžio prižiūrimas “Clock-Driven” planavimas

- Tai seniausias taikomas algoritmas. Ranka sudaromas ***tvarkaraštis***, kuris išsaugomas atmintinėj, prieš pradedant veikt sistemai
- Planuotojas, paleidęs užduotį pereina į “miego” režimą, techninės įrangos laikrodžio mechanizmas generuoja pertraukimus, kai reikia perduot valdymą kitam procesui.
- Trūkumai:
  - Reikalauja nemažai vietos tvarkaraščio saugojimui.
  - Bet koks parametrų pasikeitimas gali reikalauti viso tvarkaraščio pakeitimo
  - Algoritmas niekaip nereaguoja į veiksmų vykdymo dinamiką.

Procesoriaus dalinimo (Processor-Sharing)

- Kiekvienai užduočiai skiriama procesoriaus laiko dalis, priklausanti nuo panaudojimo faktoriaus (ui = ci/pi),
  - pav.: jei užduočiai ui = 0.35 CPU, tai užduočiai bus skirta 35 % visų laiko kvantų tam tikrame laiko intervale.
- Laiko kvantas daromas kuo mažesnis
- kontekstų persijungimo laikai yra šio algoritmo trūkumas

Prioritetais grindžiamas (Priority-Driven) planavimas

- Prioritetai gali būti priskiriami vykdymo metu:
  - (Dynamic-Priority) arba
  - gali būti fiksuoti prieš pradedant operacijas (Fixed-Priority).
    - Esant fiksuotų prioritetų planavimui:
      - mažesnis papildomų darbų kiekis.
      - planavimo algoritmai yra įdiegiami techninės įrangos lygmeny.
      - CPU panaudojimas nėra didelis (iki 70%) kaip dinaminiam planavimui (iki 100%)

Fiksuotų prioritetų planavimo algoritmai

- Naudojami Rate-Monotonic (RM) ir Deadline-Monotonic (DM) algoritmai.
  - RM priskiria aukščiausią prioritetą užduočiai ***su mažiausiu periodu***.
  - DM priskiria aukščiausią prioritetą užduočiai ***su mažiausiu reliatyviu kritinės ribos laiku***.
  - DM ir RM yra identiški jei reliatyvus kiekvienos užduoties kritinės ribos laikas yra identiškas jos periodui
  - DM yra optimalus esant vienam procesoriui.
  - DM ir RM nėra optimaliais esant dviems ar daugiau procesorių.
  - DM veikia geriau esant atsitiktinėm reliatyviom kritinėm ribom
  - RM optimalus, kai reliatyvi kritinė riba sutampa su periodu

RM planavimas

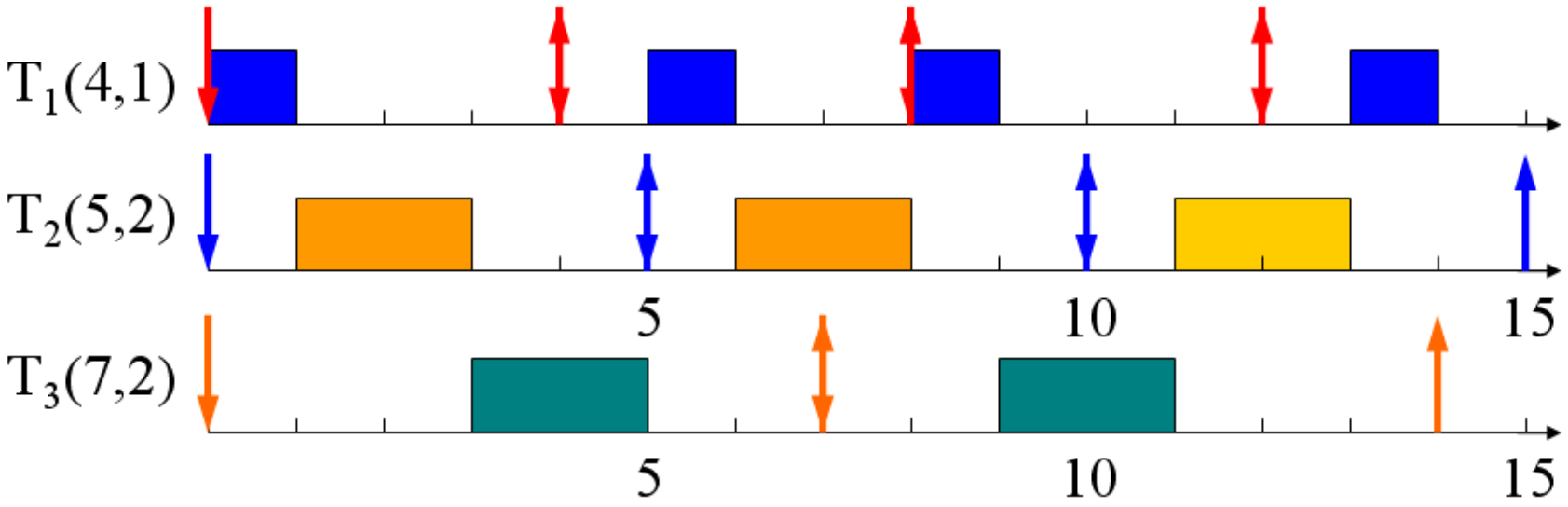
- **Planuojamumas:**
- ∑**ui** neviršija **Wn=n*(2$^{1/n}$-1)** reikšmės (Liu & Layland,
- "Scheduling algorithms for multi-programming in a hard-real-time environment", Journal of ACM, 1973). Čia **n**-užduočių kiekis sistemoje.
- Pavyzdžiui, lai turime tris procesus: T1(p1=4,c1=1), T2(p2=5,c2=1), T3(p3=10,c3=1), tada :
- $\sum Ui = 1/4 + 1/5 + 1/10 = 0.55$
- **Wn=n*(2$^{1/n}$-1)** = $3\,(2^{1/3}-1) \approx 0.78$
- Taigi, {T1, T2, T3} procesus galima surikiuoti taikant RM algoritmą.
- Augant užduočių kiekiui procesoriaus panaudojimas artėja prie 69,3%

Dinaminiai prioritetų planavimo algoritmai

Earliest-Deadline-First (EDF)

- Pagal šį algoritmą didžiausias prioritetas priskiriamas užduočiai, kurios įvykdymo **kritinė *riba yra artimiausia einamu momentu***.
- Algoritmas yra optimalus vieno procesoriaus atveju ta prasme, kad jei grupei užduočių galima realiai pritaikyti dinaminį prioritetų planavimą, tai šį planavimą galima įvykdyt naudojant EDF algoritmą.
- EDF **nėra optimalus** esant 2 ar daugiau procesorių.

EDF algoritmas



Dinaminiai prioritetų planavimo algoritmai

mažiausias neapibrėžtumas -Least Laxity

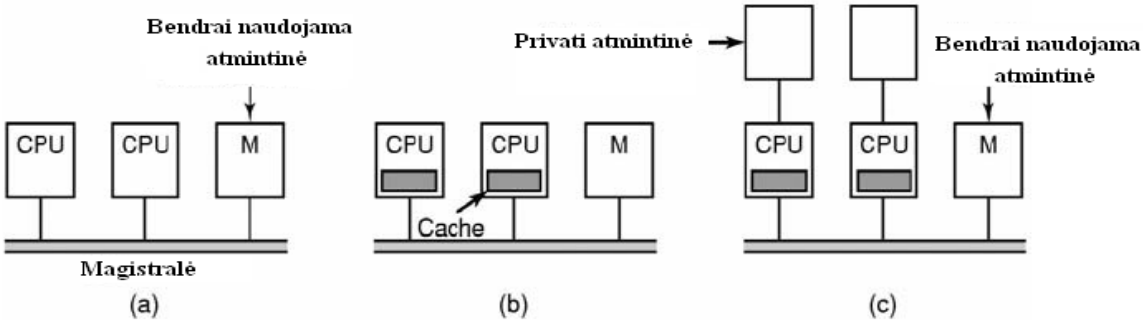
- **Planavimo politika:**
  - Visada vykdoma užduotis su mažiausia neapibrėžtumo (laxity li = di – ci) reikšme
- **Funkcionalumas:**
  - **Optimalus esant vienam procesoriui (duoda iki 100% CPU panaudojimą visose situacijose)**
    - – savo savybėmis panašus į EDF
  - **Kiek bendresnis nei EDF esant multi-procesorinei sistemai**

Planavimas daugiaprocesorinėse sistemose

- Daugiaprocesorine sistema vadinama kompiuterinė sistema, kurioje du ar daugiau CPU dalinasi prieiga prie bendros atmintinės.
- Vykdant tarpusavyje surištas užduotis atsiranda papildomi reikalavimai procesorių darbo sinchronizacijai.
- Kadangi procesoriai naudojasi bendra atmintine, neišvengiamai kyla tarpusavio išskirtinumo problema

Daugiaprocesorinių sistemų techninė įranga

- Daugiaprocesorinių kompiuterių architektūra gali būti nusakoma komponenčių sujungimo schema



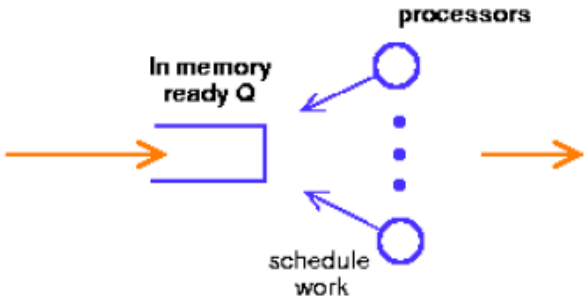
CPU sujungimas per bendrą magistralę :

a)be lokalios atmintinės, b) naudojant spartinančias atmintines, c) su lokalia atmintine

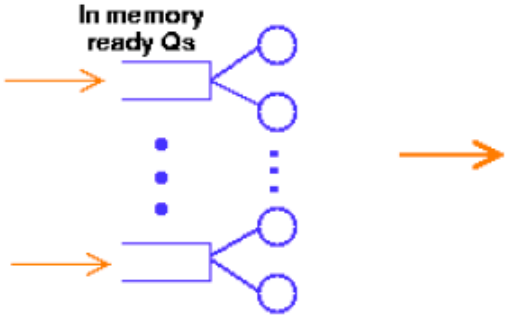
Procesų priskyrimas procesoriams

- Procesoriai gali būti traktuojami kaip laisvų resursų grupė ir procesas priskiriamas konkrečiam CPU jam pareikalavus aptarnavimo.
- Procesas gali būti “kietai” pririšamas prie tam tikro CPU
  - Taikant grupinį (gang) planavimą
  - Su kiekvienu CPU gali būti surišama eilė
  - Mažiau papildomų veiksmų
  - Vienas CPU gali būti laisvas, o kiti perkrauti

Eilės



- Viena eilė – siaura vieta



- Daug eilių – apkrovos balansas

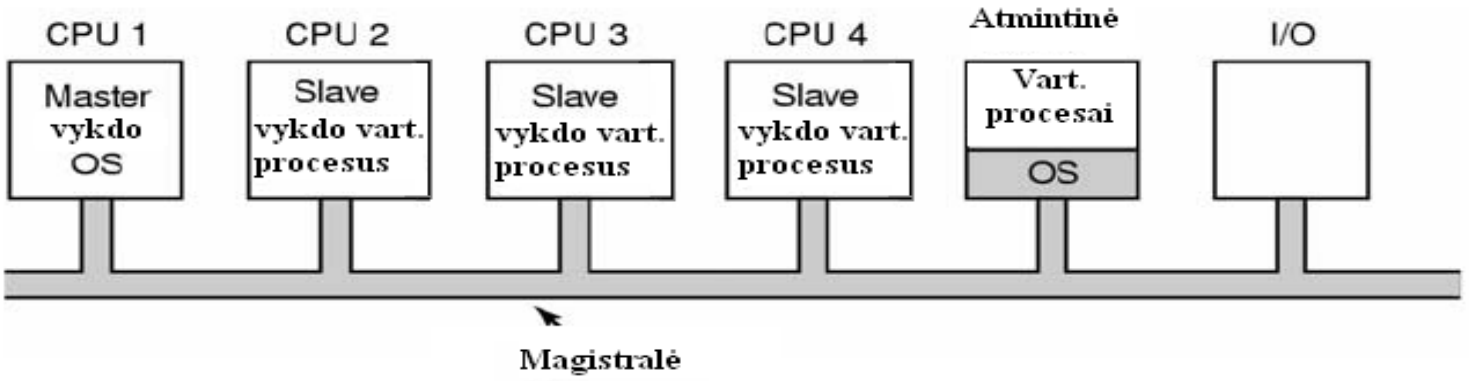
Planavimo principai

- Esant sistemoje keliems procesoriams jų tarpusavio santykiai pagal jų vykdomas funkcijas gali būti labai įvairūs.
- Galimi santykių tipai:
  - „Pono- tarno“ (Master/slave).
  - Lygiaverčiai.

„Master-slave“ tipo daugiaprocesorinė sistema

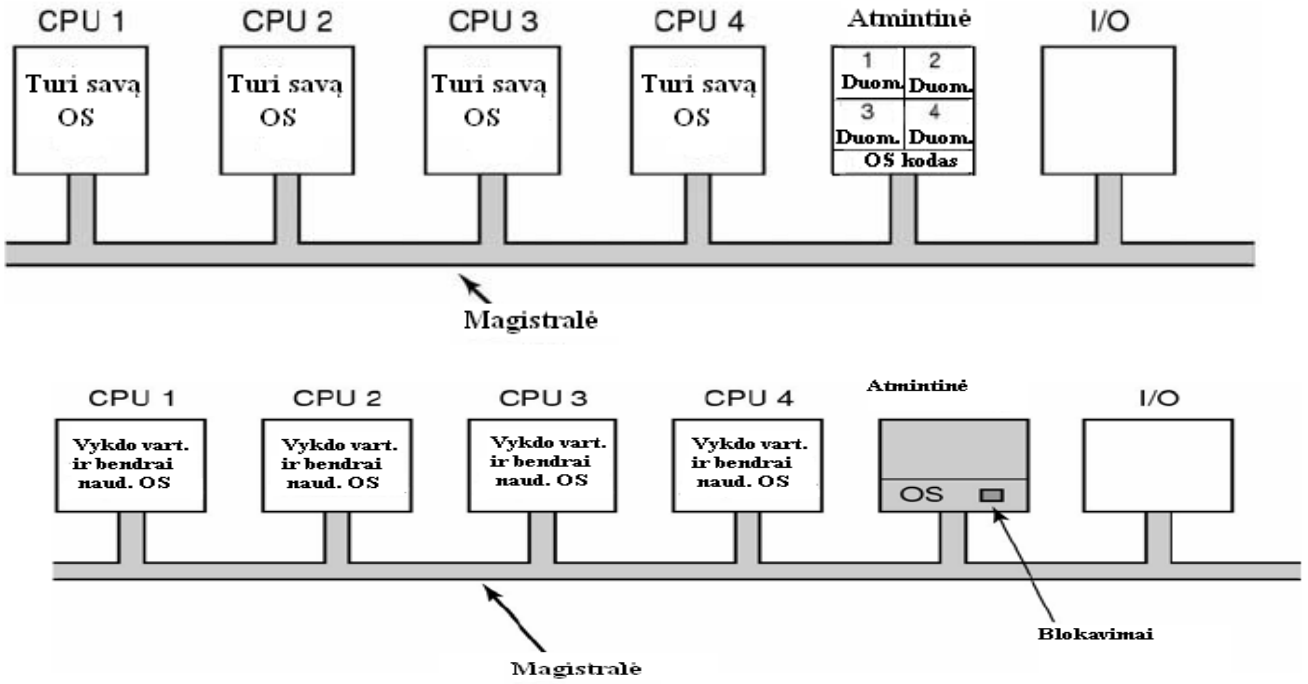
- “Master” yra atsakingas už planavimą.
- “Slave” siunčia užklausas “master” CPU.

- Minusai:
  - nulūžus „Master“ procesoriui, nulūžta visa sistema.
  - „Master“ procesorius gali tapti siaura vieta funkcionavime.

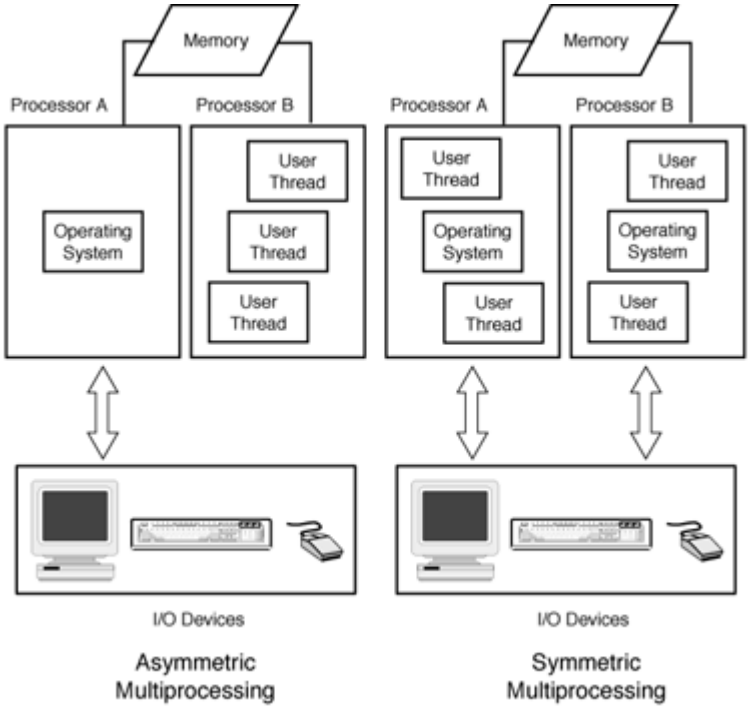


Lygiavertė (Peer) architektūra

- operacinės sistemos veiksmai gali būti vykdomi bet kuriame iš procesorių.
- Kiekvienas procesorius vykdo savą planavimą.
- Reikia tam tikros garantijos, kad du procesoriai neišrinktų to paties proceso.
- Galima nesimetrinė lygiavertė arba simetrinė lygiavertė daugiaprocesorinė sistema

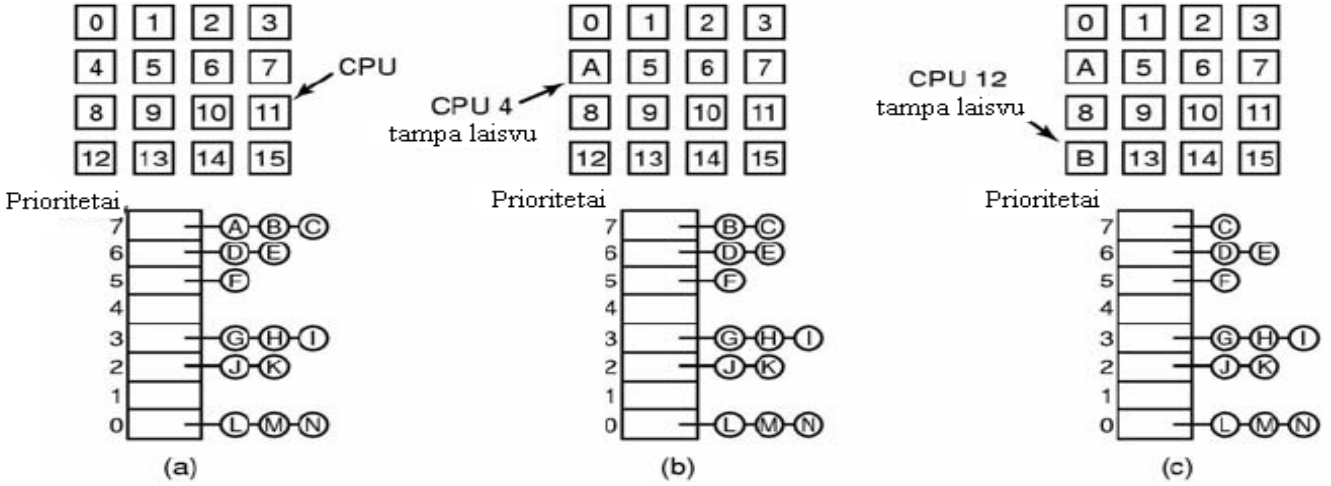


Keli procesorių branduoliai



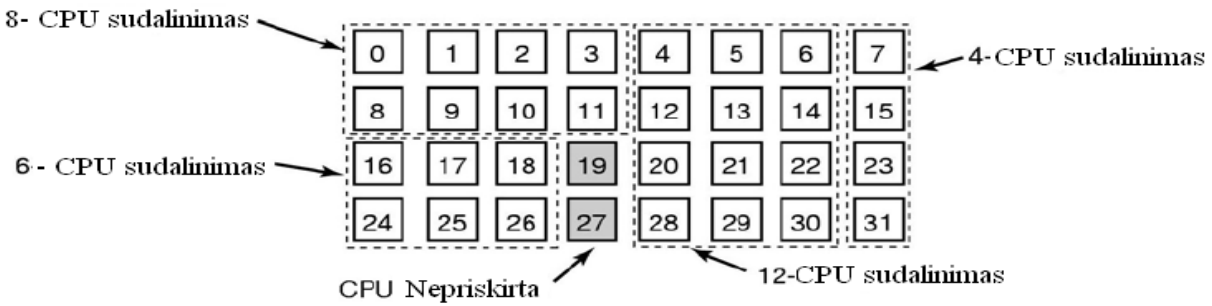
Prioritetinis aptarnavimas daugiaprocesorinėje sistemoje

- Specifinės planavimo disciplinos nėra tiek svarbios esant daugiau nei vienam procesoriui - svarbu yra užduočių paskirstymas procesoriams.
- Procesai surikiuojami į skirtingas eiles pagal savus prioritetus ir atsilaisvinus vienam iš procesorių, planuotojas ima procesą iš aukščiausio prioriteto eilės



Procesų priskyrimas procesoriams

- Galima turėti bendrą pasiruošusių procesų eilę ir priskirti procesą tuo metu esančiam laisvam procesoriui („Master-slave“ atvejas).
- Galima realizacija, kai su kiekvienu procesoriumi yra surišta pasiruošusių procesų eilė
- Pasiruošusių procesų eilė gali būti sudalinama.
- Kiek kitoks planavimas gali būti taikomas tarpusavyje susijusiems procesams Tarkim, kad vienu metu yra sukuriama grupė tarpusavyje susijusių gijų. Atsiradus šiai grupei planuotojas tikrina, ar yra tiek laisvų procesorių kiek yra tarpusavy susijusių gijų. Jei yra, tai gijoms priskiriami procesoriai ir visos jos startuoja, o jei nėra, jos visos laukia.



Daugiaprocesorinė Linux

- Turi pasiruošusių procesų eilę kiekvienam CPU
  - Kiekvienas CPU apdoroja savus procesus ir neturi laukti kol kitų CPU užduotys baigs naudot savus laiko kvantus
- void load_balance()
  - Ši funkcija bando permesti užduotis nuo vieno CPU prie kito subalansuojant CPU panaudojimus, jei tai reikalinga.
  - Kviečiama
    - Kai pasiruošusių procesų eilės nėra subalansuotos
    - Arba, periodiškai, kas kažkiek laiko
- Procesai gali būti pririšami prie tam tikro CPU.