Vilniaus universitetas
Matematikos ir informatikos fakultetas
Programų sistemų katedra
Programų sistemų II kursas, 3 grupė

„Multiprograminės operacinės sistemos projektas“

Versija 1.0

3 Laboratorinį darbą atliko:

Andrius Semionovas
Karolis Voicechovskis
Piotr Petunov

Vilnius, 2011

Turinys

1. Užduotis

1.1. Užduoties variantas

Užduočiai atlikti reikia suprojektuoti OS jūsų realiai mašinai. Operacinė sistema konstruojama kaip lygiagrečių procesų, besivaržančių dėl resursų, rinkinys, tad reikia aprašyti, kokie procesai sudaro sistemą, kas juos sukuria, kokia jų paskirtis, kaip jie veikia, kokius resursus kuria ir kokių laukia. Projekto detalumas turėtų būti toks, kad būtų aišku kaip sistema veiks. Operacinė sistema dirba realioje mašinoje ir valdo jos įrenginius, todėl turėtų būti aišku, kaip ir kada vyksta ši sąveika. Be to, esminė operacinės sistemos paskirtis yra vykdyti vartotojo užduotis - projekte turėtų matytis, kaip vartotojo programa patenka į sistemą, kaip vykdoma, kaip pateikiami rezultatai.
Atsiskaitymo metu reikės pateikti sistemos projektą (faile ar popieriuje), jį paaiškinti, argumentuoti savo sprendimus, atsakyti į klausimus apie projektuojamą OS, mokėti modifikuoti projektą. Reikalaujama išmanyti dalykinę sritį, mokėti paaiškinti, kaip projekte įgyvendinami užduoties reikalavimai.

2. Procesai

2.1. Procesų būsenos

Šiame multiprogramines operacinės sistemos modelyje (MOS modelis) yra keletas procesų būsenų. Šios būsenos turi užtikrinti sklandu procesų varžymasi dėl resursų,

Būsenos:

- Vykdomas – turi procesorių
- Blokuotas – prašo resurso (ne procesoriaus)
- Pasiruošęs – turi visus rekiamus resursus išskyrusprocesorių
- Sustabdytas – kito proceso sustabdytas procesas. Jei procesas prieš sustabdymą buvo būsenoje “Pasiruošęs”, tai jis tampa “Pasiruošęs sustabdytas”, jei buvo būsenoje “Blokuotas” - “Blokuotas sustabdytas”.

Remsimės šią procesų būsenų sąveikos modeliu (1 pav.)

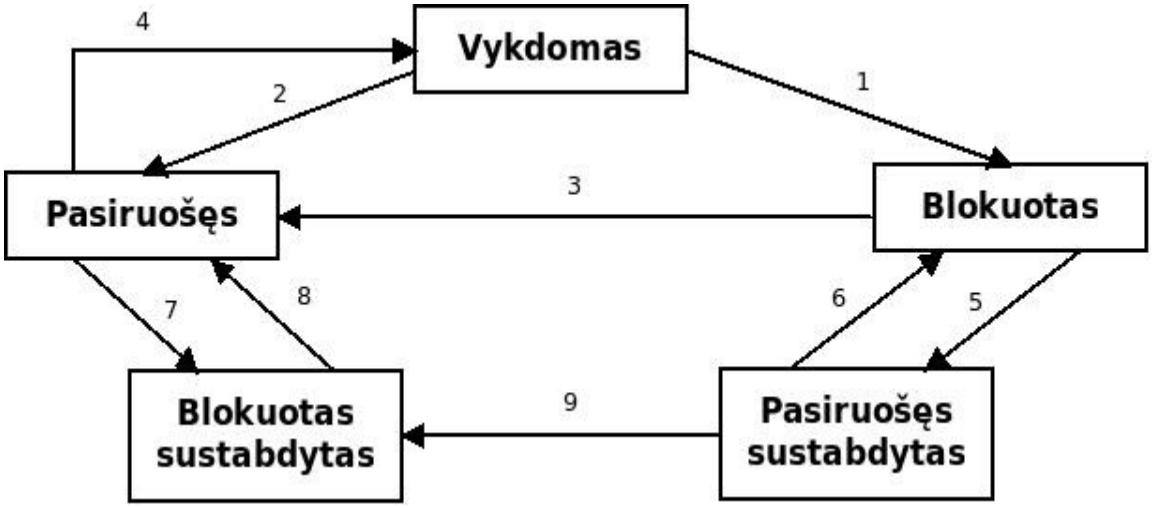


Diagrama 1: Būsenų sąveika

Naudojamame būsenų sąveikos modelyje yra devyni perėjimai:

1. Vykdomas procesas blokuojasi jam prašant ir negavus resurso.
2. Vykdomas procesas tampa pasiruošusiu atėmus iš jo procesorių dėl kokios nors priežasties (išskyrus resurso negavimą).
3. Blokuotas procesas tampa pasiruošusiu, kai yra suteikiamas reikalingas resursas.
4. Pasiruošę procesai varžosi dėl procesoriaus. Gavęs procesorių procesas tampa vykdomu.
5. Procesas gali tapti sustabdytu blokuotu, jei einamasis procesas jį sustabdo, kai jis jau ir taip yra blokuotas.
6. Procesas tampa blokuotu iš blokuoto sustabdyto, jei einamasis procesas nuima būseną sustabdytas.
7. Procesas gali tapti pasiruošusiu sustabdytu, jei einamasis procesas jį sustabdo, kai jis yra pasiruošęs.
8. Procesas tampa pasiruošusiu iš pasiruošusio sustabdyto, jei einamasis procesas nuima būseną sustabdytas
9. Procesas tampa pasiruošusiu sustabdytu iš blokuoto sustabdyto, jei procesui

yra suteikiamas jam reikalingas resursas.

2.2. Planuotojas

Planuotojas (tam tikras algoritmas) šiame MOS modelyje koordunuos procesų darbą siekiant:

- Išlaikyti optimalu procesoriaus apkrovimą, taip maksimizuojant darbų kiekį per laiko vienetą ir minimizuojant atsako vartotojui laiką;
- Po lygiai padalinti procesoriaus naudojimo laiką procesams bei koordinuoti pačių procesų vykdymą.

Planuotojas remsis prioritetais.

2.3. Prioritetai

Kiekvienas procesas turės savo prioritetą, skaičiaus reikšmę intervale nuo 1 iki 100, kur 1 reiškia žemiausią prioritetą, o 100 – aukščiausią.

Prioritetai pasitarnaus kuriant procesų sąrašus, kur aukštesniojo prioriteto procesas visada užims vietą artimesnę procesų sąrašo pradžiai.

Sisteminiai procesai visada turės aukštesni prioritetą, nei vartotojo.

2.4. Planuotojo veiksmų seka

Planuotojo darbo schema pavaizduota 2 diagramoje.

Pirmas planuotojo žingsnis yra patikrinti einamojo proceso būseną. Jei jis yra blokuotas, tai jis turi būti įtrauktas į blokuotų procesų sąrašą, jei ne – pasiruošusių sąrašą. Suradus procesą su didžiausiu prioritetu (nagrinėjamas procesas) reikia sužinoti ar egzistuoja laisvas procesorius, kuris vykdytų nagrinėjamą procesą. Radus laisvą procesorių jis yra perduodamas nagrinėjamam procesui, kuris tampa vykdomu (planuotojo darbas baigtas), jei laisvo procesoriaus nėra, reikia surasti vykdomą procesą su mažiausiu prioritetu (vykdomas procesas) ir palyginti jo prioritetą su nagrinėjamo proceso. Nustačius, kad vykdomo proceso prioritetas yra aukštesnis, planuotojas baigs darbą, jei gauname priešingą atvejį prieš atimant procesorių iš vykdomojo proceso dar reikia patikrinti ar vykdomas procesas nėra pats

planuotojas. Esant situacijai, kai vykdomas procesas – planuotojas reikia nagrinėjama procesą įsiminti ir planuotojo darbo paskutiniame žingsnyje perduoti planuotojo procesorių nagrinėjam procesui.

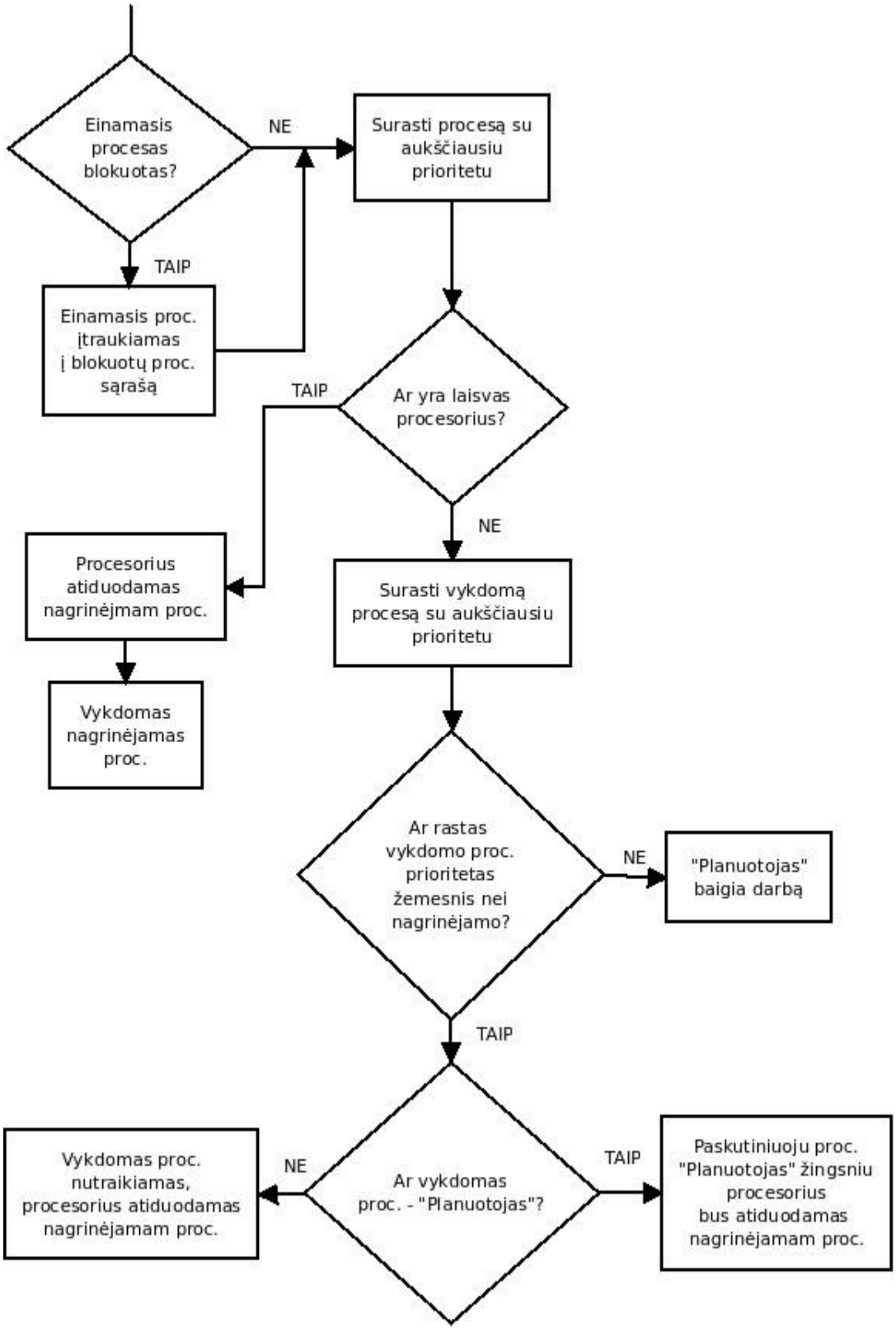


Diagrama 2: Planuotojo veiksmų seka

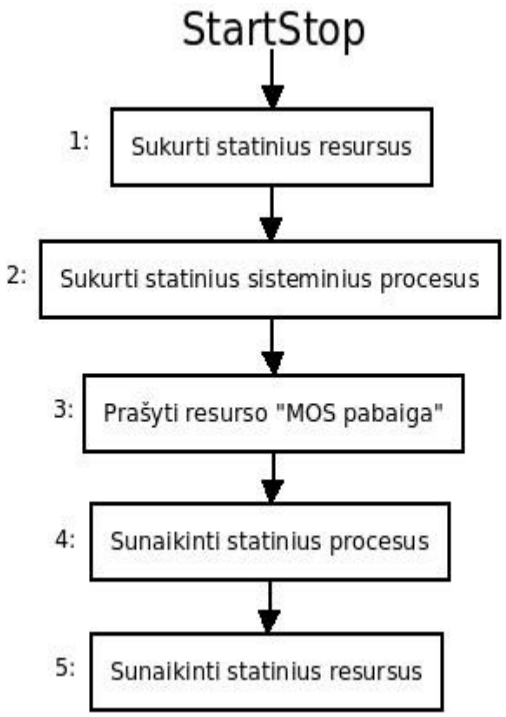


Diagrama 3: StartStop

4.2. Procesas “Read”

Šis procesas sukuriamas būsenoje „Blokuotas“. Proceso “Read” darbo eiga(4 diagrama). Ši procesa kuria ir naikina procesas „StartStop“. Proceso paskirtis – gavus informacija iš įvedimo srauto nukopijuoti užduoties kodą į supervizorinę atmintį.

Proceso žingsniai:

1. Procesas išlieka blokuotas kol negaus resurso „Įvedimas iš vartotojo aplinkos“.
2. Prašoma išskirti atminties supervizoriaus atmintyje ({atminties kiekis}).
3. Kopijuojami duomenis į išskirtą atminties buferį.
4. Tikrinama ar visa užduotis jau nukopijuota.
5. Sukuriamas resursas „Užduotis supervizoriaus atmintyje“.

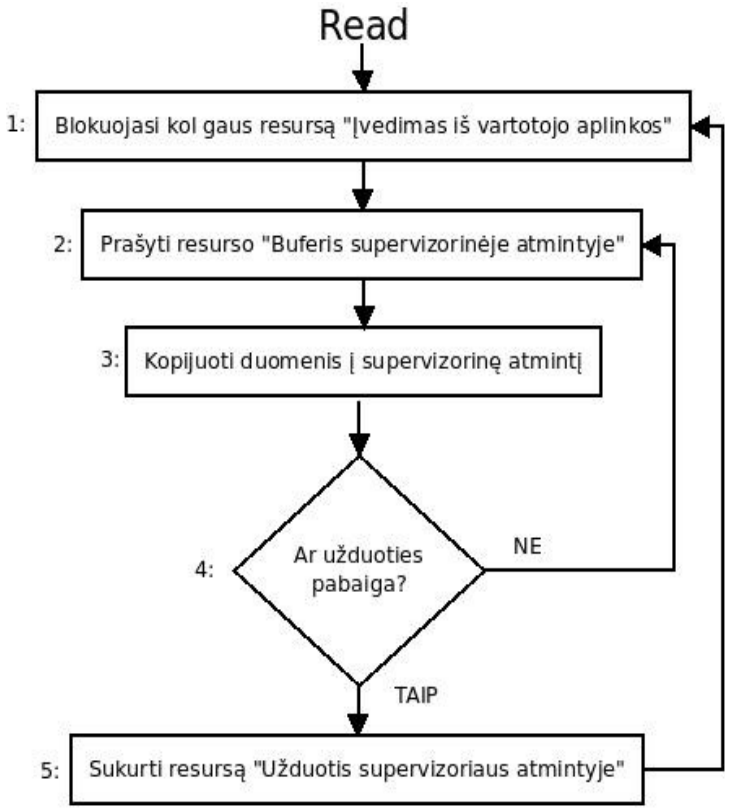


Diagrama 4: Read

4.3. Procesas „JCL“

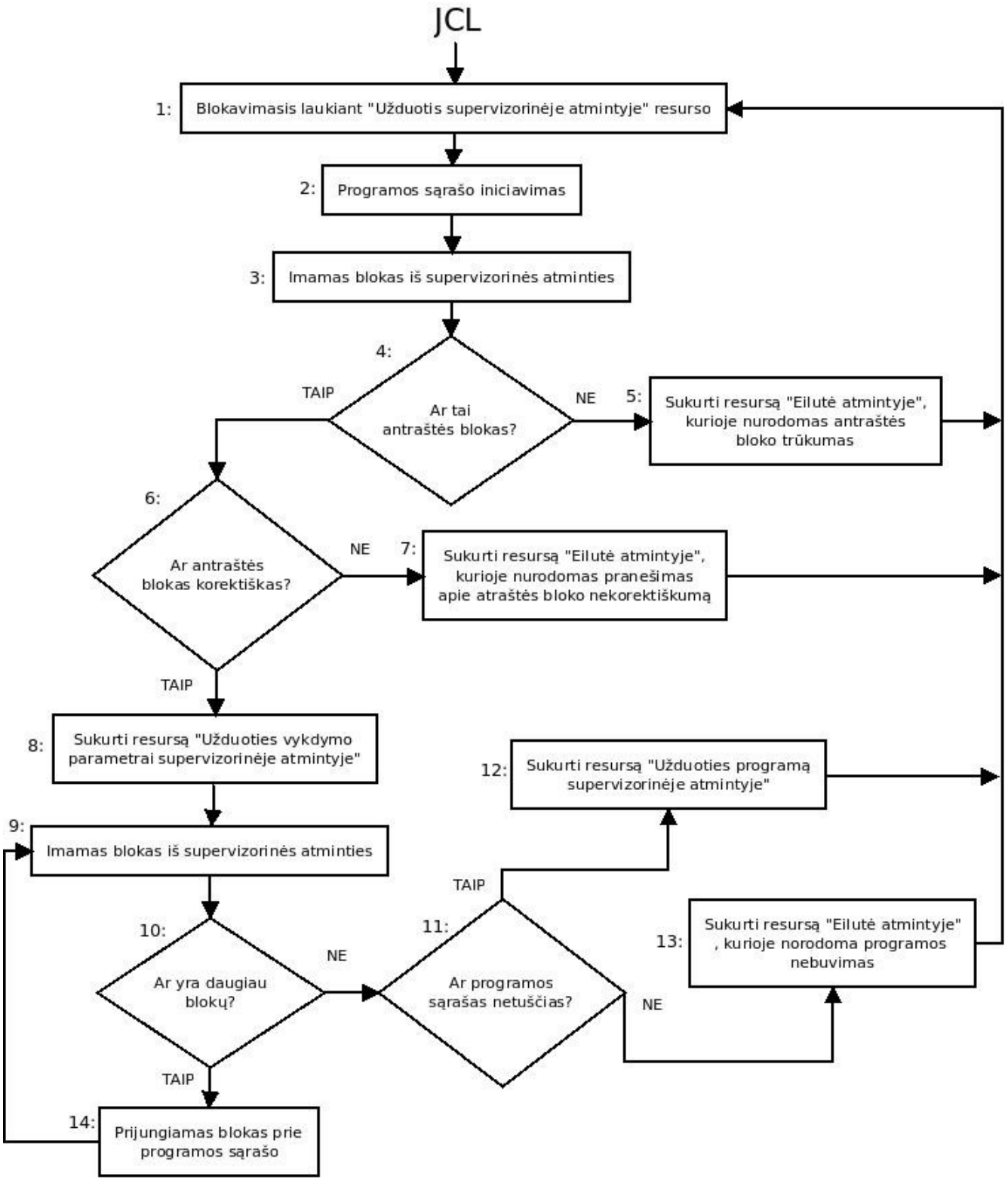
Procesą „JCL“ kuria ir naikina procesas „StartStop“. Proceso paskirtis – gautus atminties blokus iš proceso „Read“ suskirstyti į antraštės blokus ir programos blokus.

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas prnešimo iš „Read“ proceso.
2. Procesas pasiruošia darbui inicijuodamas programos blokų sąrašą.
3. Imamas pirmas blokas iš supervizorinės atminties.
4. Tikriname ar paimtas blokas yra antraštės blokas.
5. Jei pirmasis blokas nėra antraštės blokas, tai sukuriamas resursas „Eilutė atmintyje“ (parametras - eilutė), kuriame pranešama apie programos antraštės bloko nebuvimą.
6. Jei pirmasis blokas yra antraštės blokas, tada toliau tikriname ar antraštės blokas korektiškas.

7. Jei antraštės blokas nėra korektiškas, tai sukuriamas resursas „Eilutė atmintyje“ (parametras – eilutė), kuriame pranešama apie programos antraštės nekorektiškumą.
8. Jei antraštės blokas – korektiškas, sukuriamas resursas „Užduoties vykdymo parametrai supervizorinėje atmintyje“.
9. Imamas sekantis blokas iš supervizorinės atminties.
10. Tikriname ar yra daugiau kodo blokų supervizorinėje atmintyje.
11. Jei visi kodo blokai prijungti prie programos sąrašo, tai tikrinama ar tas sąrašas nėra tuščias.
12. Jei programos sąrašas yra tuščias, tai sukuriamas resursas „Eilutė atmintyje“ (parametras – eilutė), kuriame pranešama apie programos blokų nebuvimą.
13. Jei programos sąrašas nėra tuščias, tai sukuriamas resursas „Užduoties programa supervizorinėje atmintyje“.
14. Jei yra daugiau kodo blokų, tai nuskaitytasis kodo blokas nuskaitytasis kodo blokas prijungiamas prie programos sąrašo ir žingsniai kartojami nuo 9 punkto.

Po atliktų žingsnių procesas „JCL“ vėl tampa blokuotas, laukdamas resurso „Užduotis supervizorinėje atmintyje“.


JCL
1:
Blokavimasis laukiant "Užduotis supervizorinėje atmintyje" resurso
2:
Programos sąrašo iniciavimas
3:
Imamas blokas iš supervizorinės atminties
4:
Ar tai antraštės blokas?
TAIP
NE
5:
Sukurti resursą "Eilutė atmintyje", kurioje nurodomas antraštės bloko trūkumas
6:
Ar antraštės blokas korektiškas?
NE
7:
Sukurti resursą "Eilutė atmintyje", kurioje nurodomas pranešimas apie atraštės bloko nekorektiškumą
TAIP
8:
Sukurti resursą "Užduoties vykdymo parametrai supervizorinėje atmintyje"
9:
Imamas blokas iš supervizorinės atminties
10:
Ar yra daugiau blokų?
NE
11:
Ar programos sąrašas netuščias?
TAIP
12:
Sukurti resursą "Užduoties programą supervizorinėje atmintyje"
NE
13:
Sukurti resursą "Eilutė atmintyje" , kurioje norodoma programos nebuvimas
TAIP
14:
Prijungiamas blokas prie programos sąrašo

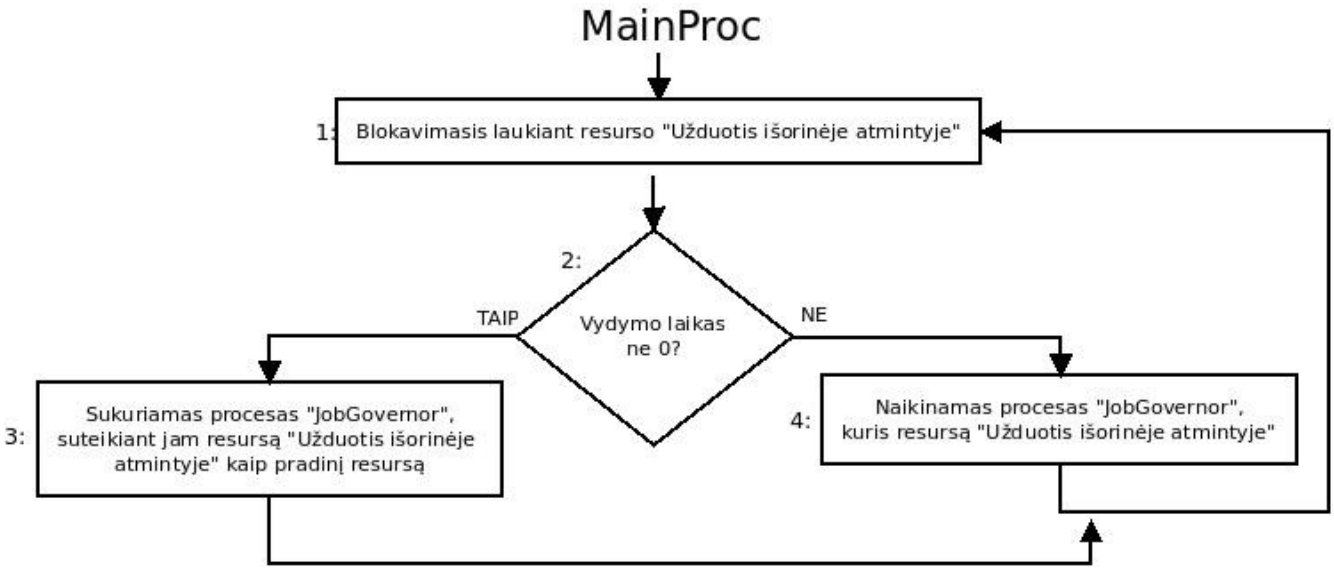
4.4. Procesas „MainProc“

Procesą „MainProc“ kuria ir naikina proesas „StartStop“. Proceso paskirtis – kurti bei naikinti procesus „JobGovernor“.

Proceso žingsniai

1. Procesas blokuojasi, laukdamas resurso „Užduotis išorinėje atmintyje“, kuri sukuria procesas „JobToDisk“.
2. Gavęs resursą, „MainProc“ tikrina ar resurso užduoties vykdymo laikas nėra lygus nuliui.
3. Jei užduoties vykdymo laikas nėra lygus nuliui, tai sukuriamas procesas „JobGovernor“, kuriam perduodamas resursas „Užduotis išorinėje atmintyje“.
4. Jei užduoties vykdymo laikas yra lygus nuliui, tai naikinamas būtent tas procesas „JobGovernor“, kuris atsiuntė resursą „Užduotis išorinėje atmintyje“.

Po atliktų proceso „MainProc“ žingsniu, jis vėl blokuojasi, laikdamas resurso „Užduotis išorinėje atmintyje“.



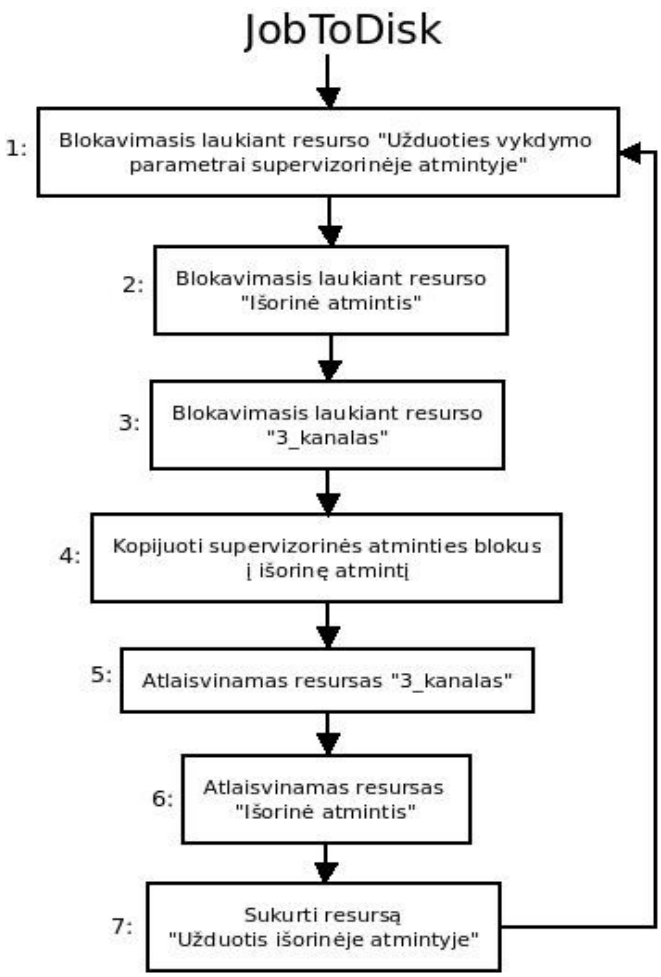
4.5. Procesas „JobToDisk“

Procesą „JobToDisk“ kuria bei naikiną procesas „StartStop“. Proceso paskirtis – perkelti užduotis blokus iš supervizorinės atminties į išorinę.

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas resurso „Užduoties vykdymo parametrai supervizorinėje atmintyje“.
2. Procesas blokuojasi, laukdamas resurso „Išorinė atmintis“.
3. Procesas blokuojasi, laukdamas trečiojo kanalo resurso „3_kanalas“.
4. Perkeliami užduoties blokai iš supervizorinės atminties į išorinę.
5. Atlaisvinamas resursas „3_kanalas“.
6. Atlaisvinamas resursas „Išorinė atmintis“.
7. Sukuriamas resursas „Užduotis išorinėje atmintyje“, kurio laukia procesas „MainProc“.

Po atliktų proceso „JobToDisk“ žingsnių, jis vėl blokuojasi, laukdamas resurso „Užduoties vykdymo parametrai supervizorinėje atmintyje“.



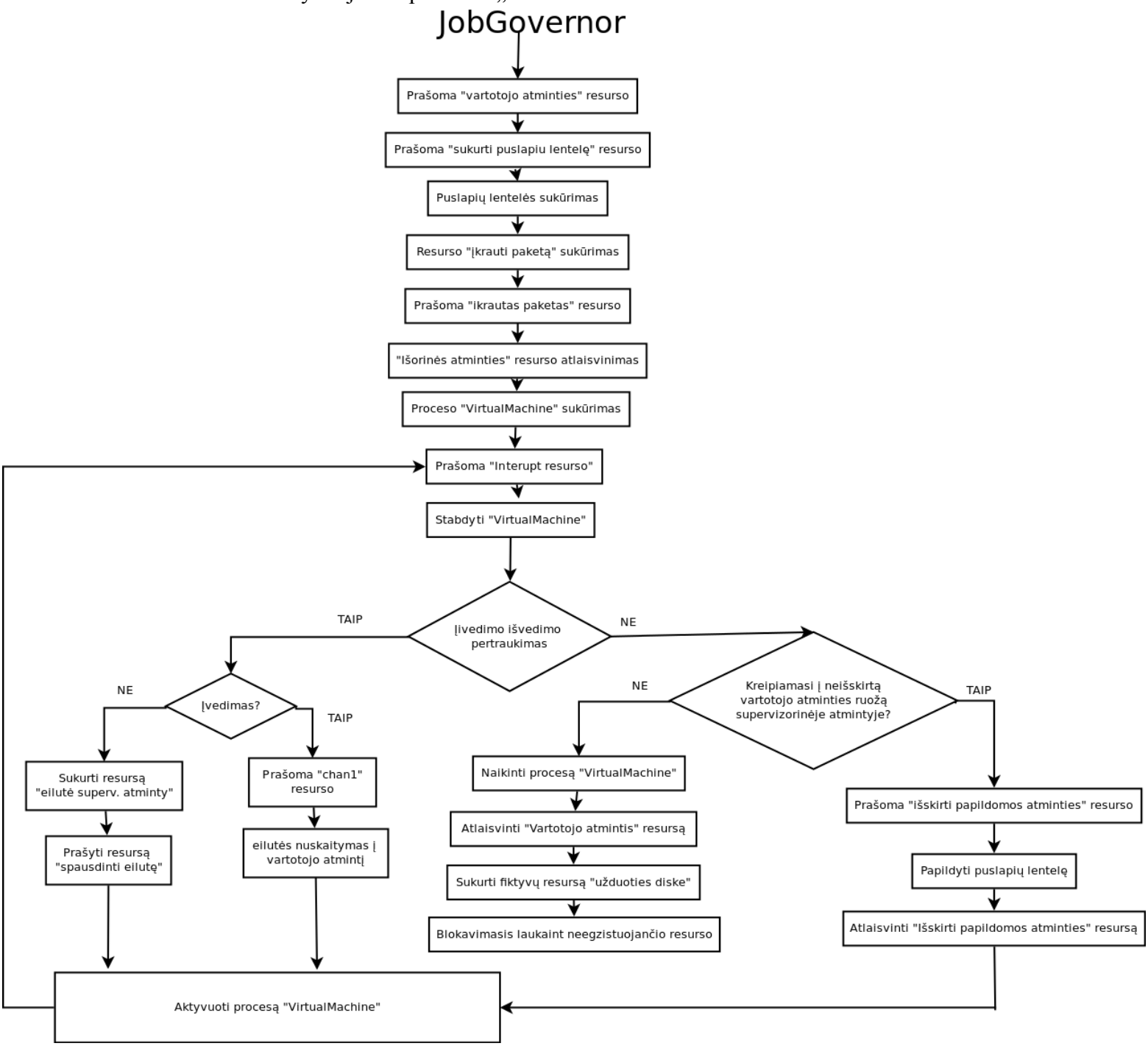
4.6. Procesas „JobGovernor“

Procesas “JobGovernor” sukuria, prižiūri bei užbaigia VirtualMachine procesą bei reaguoja į pertraukimus.

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas resurso „Vartotoji atmintis“.
2. Procesas blokuojasi, laukdamas resurso „Sukurti puslapių lentelę“.
3. Sukuriama puslapių lentelė.
4. Sukuriamas resursas „Įkrauti paketą“, kurio laukia procesas „Loader“.
5. Procesas blokuojasi, laukdamas resurso „Įkrautas paketas“.
6. Atlaisvinamas resursas „Išorinė atmintis“.
7. Sukuriamas procesas „VirtualMachine“.
8. Procesas blokuojasi, laukdamas resurso „Interrupt“.
9. Stabdomas procesas „VirtualMachine“.
10. Jei įvyko įvedimo išvedimo pertraukimas, tai tikrinama ar pertraukimas įvedimo.
11. Jei pertraukimas nėra įvedimo, tai sukuriamas resursas „Eilutė superv. atminty“.
12. Procesas blokuojasi, laukdamas resurso „Spausdinti eilutę“.
13. Aktyvuojamas procesas „VirtualMachine“.
14. Jei pertraukimas yra įvedimo, tai procesas blokuojasi, laukdamas resurso „Chan1“.
15. Eilutė nuskaitoma į vartotojo atmintį.
16. Aktyvuojamas procesas „VirtualMachine“.
17. Jei įvyko ne įvedimo išvedimo pertraukimas, tai tikrinama ar kreipiamasi į neišskirtą vartotojo atminties ruožą supervizorinėje atmintyje.
18. Jei nėra kreipiamasi į neišskirtą vartotojo atminties ruožą supervizorinėje atmintyje, tai naikinamas procesas „VirtualMachine“.
19. Atlaisvinamas resursas „Vartotojo atmintis“.
20. Sukuriamas fiktyvus resursas „Užduotis diske“.
21. Procesas blokuojasi, laukdamas neegzistuojančio resurso.
22. Jei yra kreipiamasi į neišskirtą vartotojo atminties ruožą supervizorinėje atmintyje, tai procesas blokuojasi, laukdamas resurso „Išskirti papildomos atminties“.
23. Papildoma puslapių lentelė.
24. Atlaisvinamas resursas „Išskirti papildomos atminties“.

25. Aktyvuojamas procesas „VirtualMachine“.



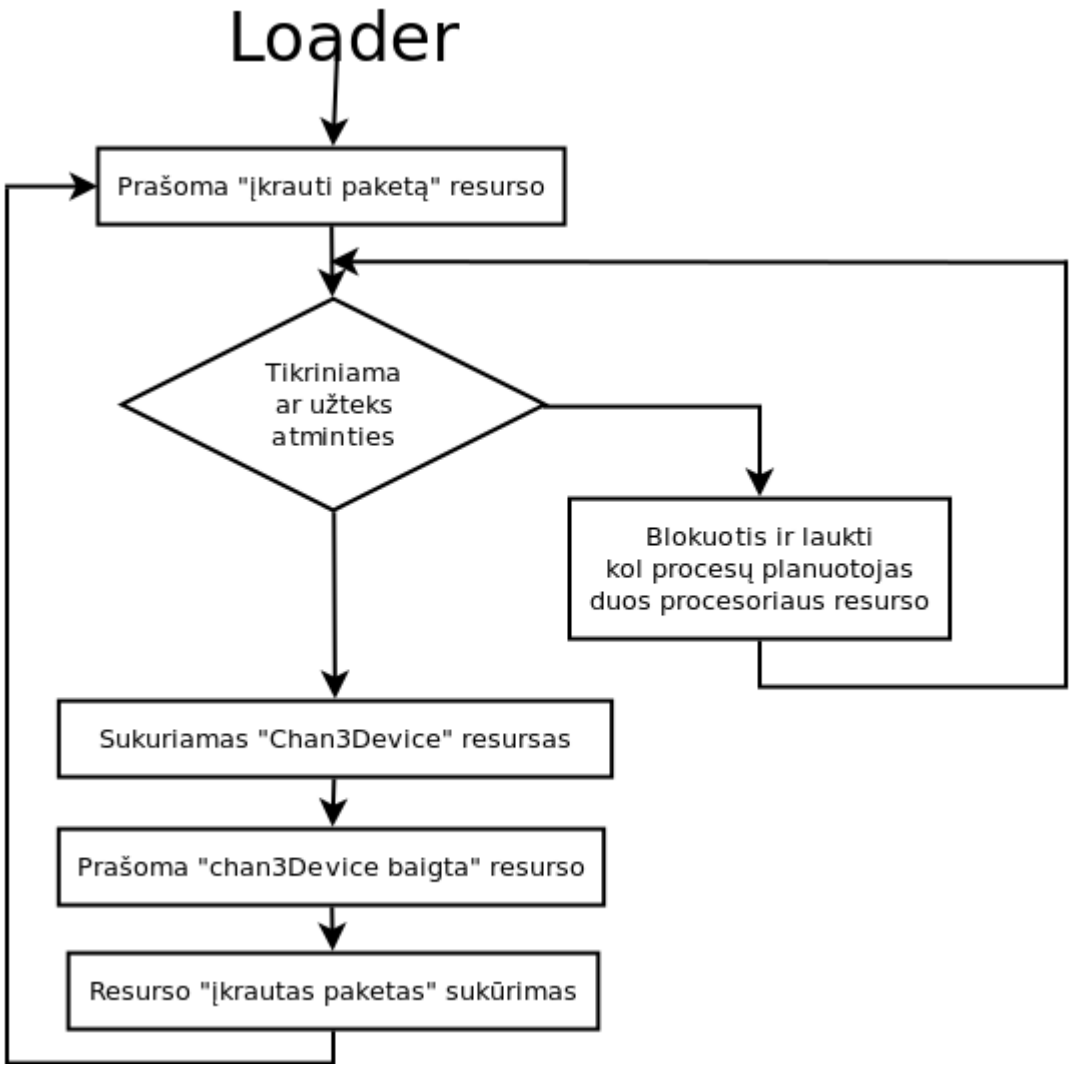
4.7. Procesas „Loader“

Procesas „Loader“ skirtas pakrauti informacijai iš disko. Šis procesas vienintelis valdo trečią kanalą.

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas resurso „Įkrauti paketą“.
2. Jei atminties paketui įkrauti neužteks, tai procesas blokuojasi, kol procesu planuotojas neduos procesoriaus resurso.
3. Jei atminties užteks, tai sukuriamas resursas „Chan3Device“.
4. Procesas blokuojasi, laukdamas resurso „Chan3Device baigta“.
5. Sukuriamas resursas „Įkrautas paketas“.

Po atliktų žingsnių procesas „Loader“ vėl tampa blokuotas, laukdamas resurso „Įkrauti paketą“.



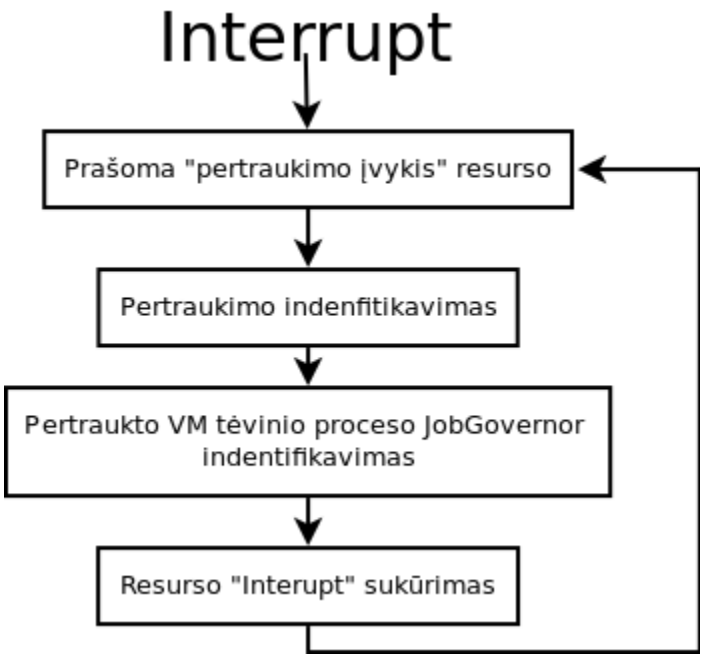
4.8. Procesas „Interrupt“

Procesas “Interrupt” reaguoja tiek į Virtualios mašinos pertraukimus, tiek į išorinius pertraukimus (pastarųjų pertraukimo resursą sukuria branduolio primityvas).

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas resurso „Pertraukimo įvykis“.
2. Identifikuojamas pertraukimas.
3. Identifikuojamas VM tėvinis procesas JobGorvernor.
4. Sukuriamas resursas „Interrupt“.

Po atlinktų žingsnių procesas „Interrupt“ vėl tampa blokuotas, laukdamas resurso „Pertraukimo įvykis“.



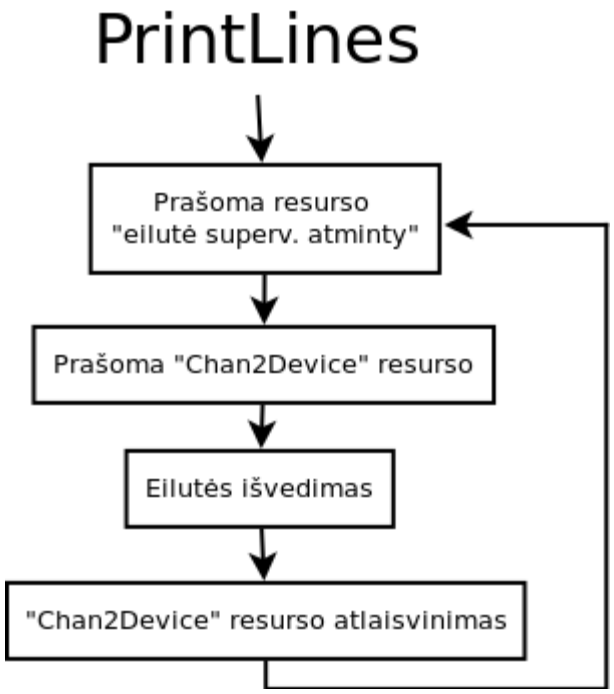
4.9. Procesas „PrintLines“

Procesas “PrintLines” yra skirtas išspausdinti eilutę.

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas resurso „Eilutė superv. atminty“.
2. Procesas blokuojasi, laukdamas resurso „Chan2Device“.
3. Išvedama eilutė.
4. Atlaisvinamas resursas „Chan2Device“.

Po atlinktų žingsnių procesas „PrintLines“ vėl tampa blokuotas, laukdamas resurso „Eilutė superv. atminty“.



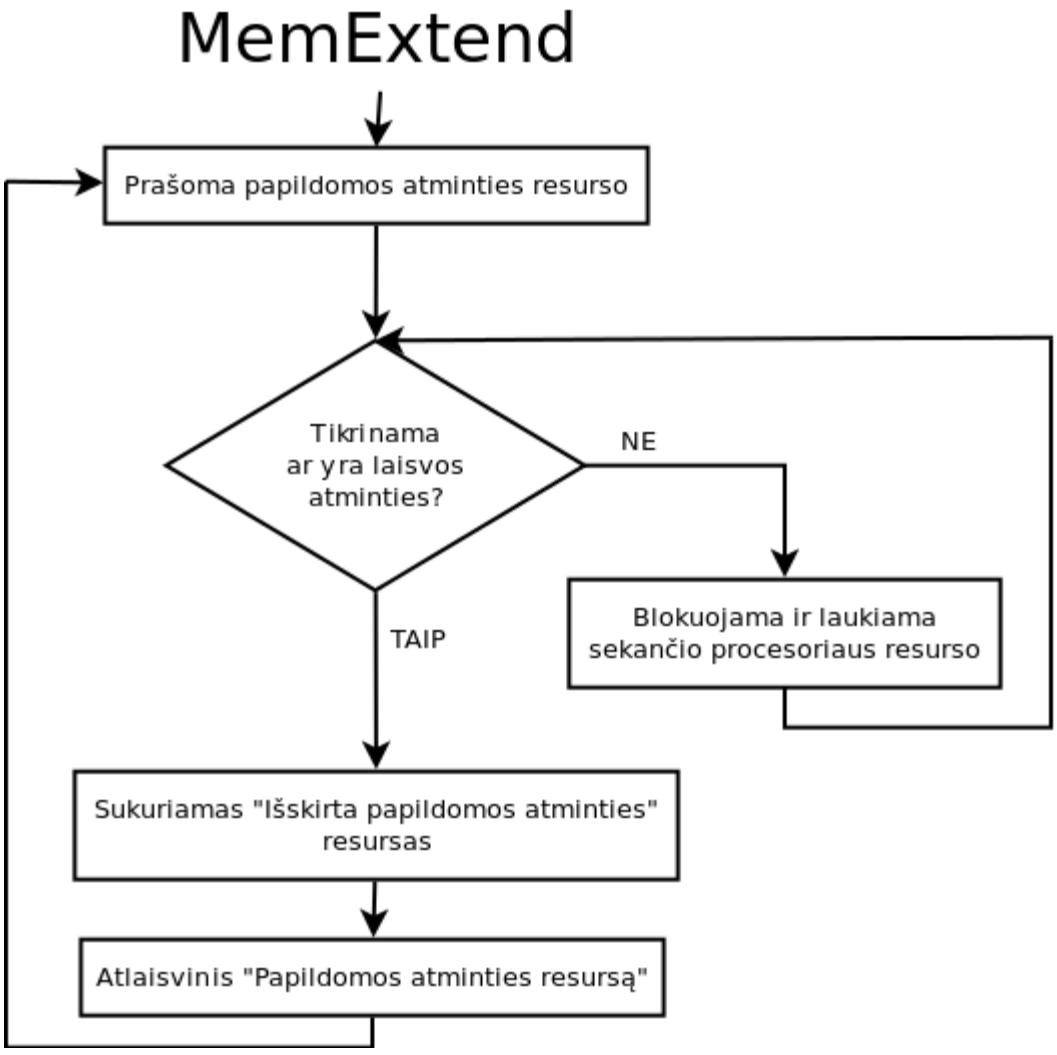
4.10. *Procesas „MemExtend“*

Procesas “MemExtend” yra skirtas rasti ir išskirti laisvos atminties virtualiai mašinai. Jeigu atminties nėra, tuomet procesas turi laukti kol atsiras reikalingo dydžio atmintis ir neleisti virtualiai mašiniai aktivuotis. Tai užtikrina resurso “Išskirta papildomos atminties” resurso sukūrimas tik tuomet, kada atmintis išskirta.

Proceso žingsniai:

1. Procesas blokuojasi, laukdamas resurso „Papildomos atminties resursas“.
2. Jei nėra laisvos atminties, tai procesas blokuojasi, laukdamas kito proceso resurso.
3. Jei yra laisvos atminties, tai sukuriamas resursas „Išskirta papildomos atminties“.
4. Atlaisvinamas resursas „Papildomos atminties resursas“.

Po atliktų žingsnių procesas „MemExtend“ vėl tampa blokuotas, laukdamas resurso „Papildomos atminties resursas“.



5. Deskriptoriai

Deskriptoriai yra atminties sritys skirtos saugoti inforamcija tiek apie procesus, tiek apie resursus.

5.1. Proceso deskriptorius

Id: String[16] //išorinis vardas, turi būti unikalu

CPU: int //rodyklė, kuri rodo į sisteminį deskriptorių

R: LinkedList { //turimų resursų sąrašas

resource: int //adresas rodantis į pirmąjį resursą

}

ST: int /* Skaičiumi nusakoma proceso būsena: {

VYKDOMAS = 0;

PASIRUOŠĘS = 2;

BLOKUOTAS = 3;

PASIRUOŠĘS_SUSTABYTAS = 4;

BLOKUOTAS_SUSTABDYTAS = 5;

} */

T: int //tėvinio proceso sąrašas

S: LinkedList { //sūnų sąrašas

procid: int

}

PR: int //proceso prioritetas

5.2. *Sisteminis proceso deskriptorius*

ic: int	//Instrukcijos skaitliukas
sf : int	//Status flag
s: int	//Steko viršūnė
mode: int	//Režimas (super, user)
pd: int	//Proceso deskriptorius
m: int	//Registras M
r: int	//Registras R
ptr: int	//Puslapių lentelės registras

Pagrindinio proceso desktirpius visada yra pirmuose 8 žodžiuose atmintyje, ir jį sudaro visi registrai (įskaitant ir pilkus). Visi kiti procesai yra aprašomi tik juodai pažymėtais registrais ir jie gali būti bet kurioje atminties vietoje.

5.3. *Resurso deskriptorius*

RID: String[16]	//resurso išorinis vardas
PNR: boolean	//pakartotinio panaudojamumo resursas?
KID: int	//sukūrurio proceso ID
LPS: Queue {	//resurso laukiančių procesų sąrašas
procid: int	//proceso ID
priority: int	//prioritetas
}	
PASK: int	//paskirstymo funkcijos adresas

6. Primityvai

Primityvai yra OS branduolio dalis, kuri nėra atskiri procesai, o yra vykdomi kaip paprogramės juos kviečiančių procesų aplinkoje. Primityvams argumentai perduodami per steką (kaip ir visoms kitoms paprogramėms). Turime šiuos primityvus:

1. **Darbui su mašina:**
    1. **INNER_INT** - Iškviesti vidinį pertraukimą
    2. **GET_MEM** - Gauti reikiamą laisvų segmentų kiekį superv. Atmintyje
    3. **EXT_MEM** - Prašyti papildomos atminties.
    4. **MAKE_PT** - Suformuoti puslapių lentelę
2. **Darbui su duomenimis:**
    1. **PUSH_LL** - Papildyti LinkedList
    2. **PUSH_QL** - Papildyti QueueList
    3. **POP_LL** - Paimti ir pašalinti seniausiai įterptą įrašą iš LinkedList
    4. **PUSH_LL** - Paimti ir pašalinti elementą iš QueueList su didžiausiu prioritetu.
    5. **SET_PRIORITY** - Pakeisti prioritetą pagal programos ID
3. **Darbui su procesais**
    1. **MAKE_P** - Kurti procesą
    2. **KILL_P** - Naikinti procesą
    3. **STOP_P** - Stabdyti procesą
    4. **RUN_P** - Aktyvuoti procesą
4. **Darbui su resursais**
    1. **MAKE_R** - Kurti resursą
    2. **KILL_R** - Naikinti resursą
    3. **ASK** - Prašyti resurso
    4. **RELEASE** - Atlaisvinti resursą