VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Kursinis darbas

Modelinė operacinė sistema

Atliko: 3 kurso, 4 grupės studentai

Povilas Panavas (parašas)

Aurelijus Rožėnas (parašas)

Darbo vadovas:

doc. dr. Antanas Mitašiūnas (parašas)

Vilnius
2008

Turinys

Įžanga

Kiekvienais metais studentai rašo savo operacines sistemas. Problematiška vienu metu aprėpti didelį teorinės informacijos kiekį ir jį praktiškai panaudoti. Šio darbo tikslas yra parengti mokomąją operacinę sistemą, kurią būtų galima pateikti kaip pavyzdį praktinių užsiėmimų metu.

Darbui keliami šie uždaviniai:

1. Pateikti esminę teorinę medžiagą, reikalingą operacinės sistemos veikimo principų suvokimui.
2. Parengti modelinės operacinės sistemos projektą.
3. Sukurti modelinę operacinę sistemą remiantis parengtu projektu.

Darbas rašomas remiantis asmenine patirtimi, sukaupta kuriant virtualią mašiną, virtualios operacinės sistemos projektą ir jo realizaciją kurso „Operacinės sistemos“ metu. Sudėtingiausia dalis – apjungti teorines žinias su praktika, t.y. realizuoti studentų parengtą operacinės sistemos projektą.

Plačiausiai šiuo metu yra paplitusios multiprograminės operacinės sistemos, todėl buvo nuspręsta realizuoti būtent tokią sistemą. Kuriama sistema, nepasižymi jokiomis išskirtinėmis savybėmis. Tokį pasirinkimą lėmė tai, kad vienas šio darbo siekiamų tikslų yra nesudėtingas operacinės sistemos veikimo principų perpratimas ir įsisavinimas.

Beveik visos egzistuojančios operacinės sistemos yra skirtos atlikti praktiniams darbams, todėl nėra galimybės, stebėti jos veikimo principus ir būsenas. Taip pat, jos būna prisirišusios prie tam tikros realios kompiuterių architektūros. Norėdami apeiti šiuos du realių operacinių sistemų ypatumus, mes pasirinkome sukurti modelinę operacinę sistemą.

Darbo pradžioje pateikiamos priežastys lėmusios šios mokymo metodikos pasirinkimą, vėliau pateikiamos sąvokos, reikalingos suprasti dėstomą teoriją apie operacinių sistemų veikimo principus. Po to apibrėžiama aparatūrinė įranga – mašina, kuri bus paprasta ir leis lengviau įsisavinti operacinės sistemos veikimo principus.

1 Operacinių sistemų kurso dėstymo metodikos

Operacinių sistemų kursas yra plačiai naudojama priemonė visame pasaulyje studentų žinioms kompiuterių moksluose gilinti, todėl yra išbandyta daugybė būdų, siekiant kuo didesnio žinių įsisavinimo ir sistemos veikimo principų suvokimo. Galima išskirti tris pagrindines metodikas:

1. Naudotis jau parengtais aparatūrinės įrangos simuliatoriais, kurie gali būti:
    a. Supaprastinta aparatūrinė įranga – panaši į realią ir plačiai taikomą, tačiau lengviau perprantama studentams;
    b. Programinė įranga – imituojanti kompiuterio darbą.
2. Keisti egzistuojančios operacinės sistemos modulį arba tam tikrą algoritmą. Šiuo atveju dažnai naudojama operacinė sistema „Mynix“.
3. Kurti operacinę sistemą nuo nulio. Galimi variantai:
    a. Rašyti sistema realiai aparatūrinei įrangai;
    b. Apsibrėžti savo aparatūrinę įrangą, kurios darbą simuliuoja operacinė sistema.

Mes pasirinkome pastarąjį variantą (3.b.) dėl šių priežasčių:

- Šis metodas leidžia atsisakyti papildomų išlaidų aparatūriniai įrangai;
- Nebūtina iš anksto turėti sukurtą programinę įrangą, kuri imituoja kažkokią kompiuterio architektūrą. Be to, savo aparatūros apibrėžimas padidina studentų suvokimą;
- Realios operacinės sistemos tobulinimas turi vieną didelį trūkumą – sistemos išeities tekstų studijavimas ir kažkurios dalies keitimas bus labiau teorinis metodas, nes sunku perprasti visos sistemos darbą. Todėl tikėtina, kad tokiu būdu bus atskleista per mažai operacinės sistemos veikimo principų detalių.
- Apsibrėždami savo kompiuterio architektūrą, galime turėti paprastą architektūrą bei gauti platesnes operacinės sistemos veikimo stebėjimo galimybes. Be to, tokią operacinę sistema paleisti galima bet kuriame kompiuteryje.

2 Sąvokos

2.1 Operacinės sistemos sąvoka

Operacinę sistemą galima apibrėžti daugybe būdų. Vienas paprasčiausių būdų įsivaizduoti operacinę sistemą, kaip pagrindinę programą, po kurios užkrovimo, vykdomos visos kitos naudotojo programos.

Jei bandytume apibrėžti operacinę sistemą, remdamiesi jos atliekamais uždaviniais, tuomet tai sistema, kuri padeda paslėpti sudėtingą kompiuterio architektūrą, leidžia paprasčiau naudotis kompiuterio resursais. Taigi, operacinę sistemą galima įsivaizduoti, kaip resursus kontroliuojančią programą arba, kaip programą, pakeliančią kompiuterio architektūrą į aukštesnį lygį.

2.2 Multiprograminės operacinės sistemos sąvoka

Multiprograminė operacinė sistema (MOS) yra viena iš operacinių sistemų rūšių. Tokios operacinės sistemos leidžia lygiagrečiai vykdyti kelias programas[1]. Resursų perskirstymas tarp procesų užtrunka trumpai ir tik nežymiai padidina atskirų programų (užduočių[2]) atlikimo laiką. Be to, naudotojui vienu metu neužtenka vienos aktyvios programos, taigi MOS yra patrauklesnė paprastiems naudotojams.

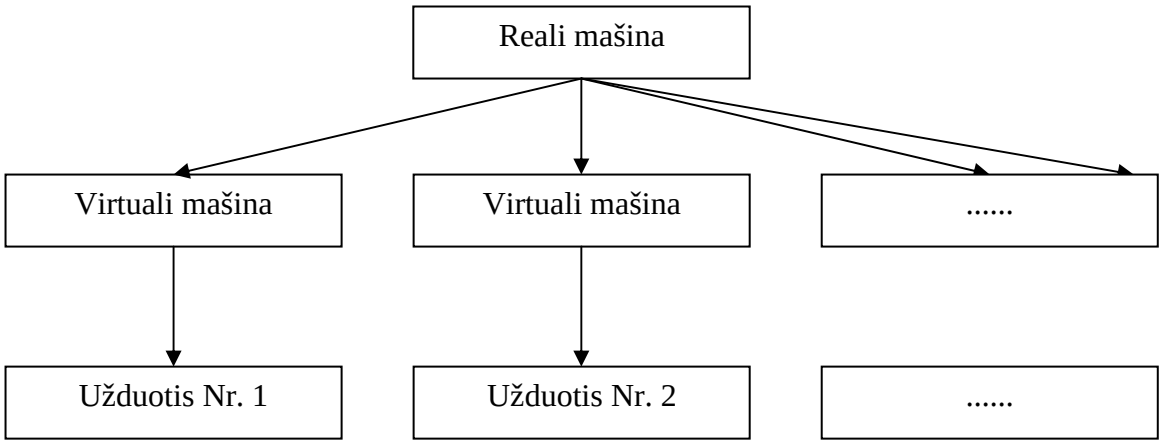
[1] Sistemoje su vienu procesoriumi vienu metu vykdoma gali būti vykdoma tik viena užduotis, tačiau vykdomos užduotys dažnai keičiamos (gauna procesoriaus resursą), todėl naudotojui susidaro įspūdis, kad vienu metu vykdomos kelios programos.

[2] Užduotimi vadiname programą su vykdymo parametrais ir startiniais duomenimis.

3 Realios ir virtualios mašinų aprašai

Reali mašina – tai kompiuteris. Reali mašina turi sudėtingą sąsają programuotojui, todėl atsiranda virtualios mašinos sąvoka. Virtuali mašina – tai tarsi supaprastinta realios mašinos kopija, siūlantį patogesnį interfeisą programuotojui.

Reali mašina vienu metu gali turėti daug virtualių mašinų (žr. Figūra 1), kurios varžosi dėl kompiuterio resursų (pačios užduotys dėl jų nesivaržo). Kiekviena virtuali mašina yra užduoties procesas, t.y. vykdo užduoties programą pagal pateiktus parametrus ir pradinius duomenis. Vadinasi, kiekviena programa sistemoje elgiasi taip, tarsi ji vienintelė būtų vykdoma, t.y. be pertraukimų.



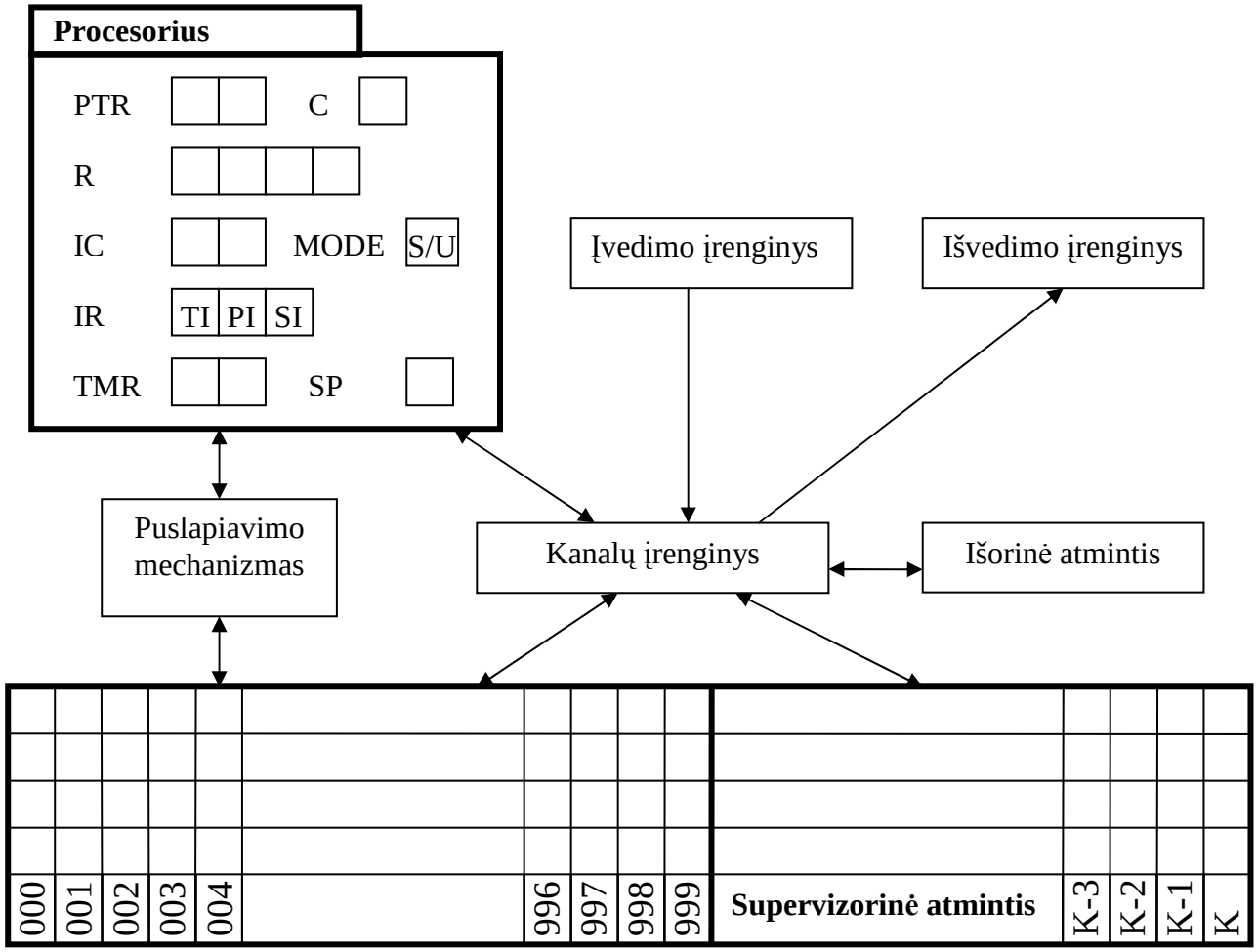
Figūra 1: Realios ir virtualios mašinų sąveika

3.1 Realios ir virtualios mašinų modeliai

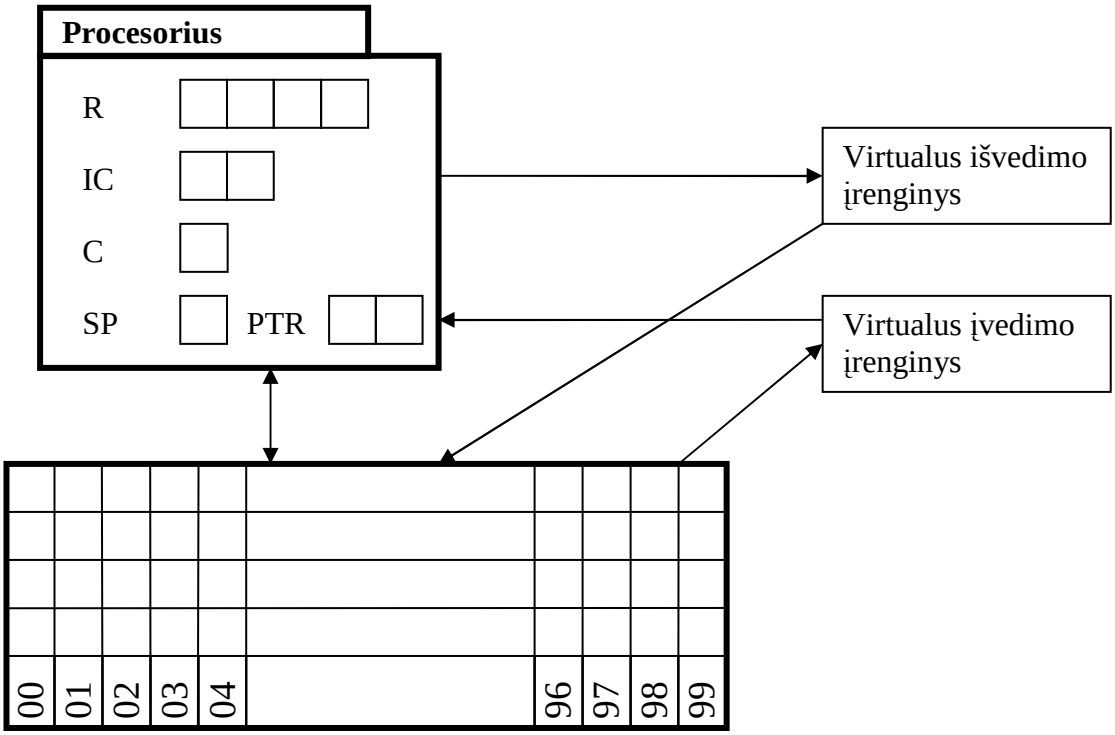
Realios mašinos modelis – tai virtualus kompiuteris. Daugelis kompiuterių architektūrų turi bendrus struktūros komponentus bei panašius veikimo principus.

Sukursime savo realios mašinos modelį (schema pateikta Figūra 2), tai leis paprasčiau pademonstruoti operacinės sistemos veikimo principus ir neprisirišti prie konkrečios kompiuterio architektūros realizacijos. Šiame darbe pateiktas vienas iš galimų realios mašinos modelių.

Virtualios mašinos modelis yra supaprastinta reali mašina, suteikianti programuotojui patogesnį interfeisą užduočių rašymui (žr. Figūra 3).



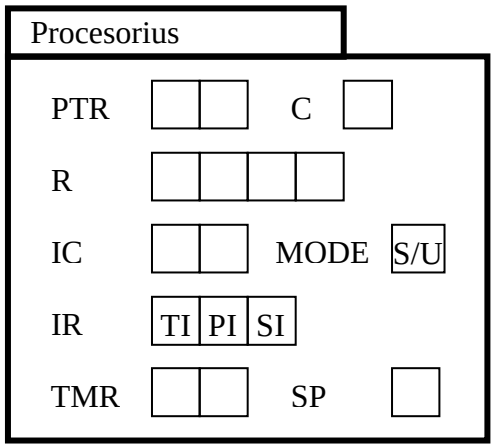
Figūra 2: Realios mašinos modelis



Figūra 3: Virtualios mašinos modelis

3.2 Procesorius

3.2.1 Realios mašinos procesorius



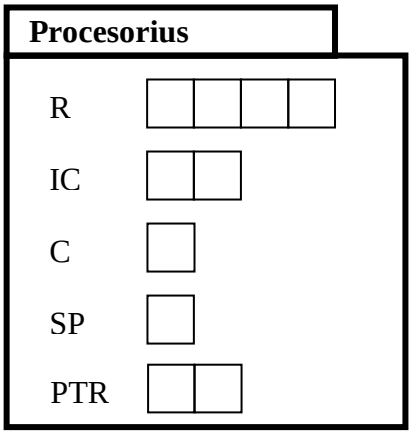
Figūra 4: Realios mašinos procesorius

Procesoriaus paskirtis - skaityti komandą iš atminties ir ją vykdyti. Procesorius gali dirbti dviem režimais: supervizoriaus ir vartotojo. Operacinė sistema veikia supervizoriniu režimu, o joje vykdoma virtualios mašinos užduotis – vartotojo režimu.

Procesoriaus registrai:

- PTR – 2 baitų puslapiavimo registras
- C – 1 baito loginis trigeris
- R – 4 baitų bendrosios paskirties darbinis registras
- IC – 2 baitų komandų skaitiklis
- MODE – 1 baito režimo registras (U – vartotojo, S – supervizoriaus režimas)
- IR – 3 baitų pertraukimų registras. Jį sudaro 3 dalys:
    - TI – taimerio pertraukimas
    - PI – programinis pertraukimas
    - SI – supervizorinis pertraukimas
- TMR – 2 baitų taimerio registras
- SP – 1 baito steko viršūnės rodyklė

3.2.2 Virtualios mašinos procesorius



Figūra 5: Virtualios mašinos procesorius

Virtualios mašinos procesorius yra apkarpyta realios mašinos procesoriaus versija. Trūkstami registrai, lyginant su realios mašinos procesoriumi, reikalingi tam, kad būtų galima organizuoti multiprograminei operacinei sistemai būdingą darbo eigą. Tai leidžia vykdomai užduočiai elgtis taip, tarsi ji būtų vienintelė visoje sistemoje.

Virtualios mašinos procesoriaus registrų paskirtis atitinka realios mašinos procesoriaus registrų paskirtį, todėl nebus detaliau aprašinėjama.

3.3 Procesoriaus komandos

Šiame skyriuje pateikiamos komandos, kurias galės vykdyti mūsų kuriamos sistemos procesorius.

Pastaba: ST – steko elementas, pvz., ST[2] reiškia antrą steko elementą. SP rodo į steko viršūnę.

3.3.1 Aritmetinės komandos

1. ADxy – sudeda žodžio patalpinto adresu xy reikšmę su registru;
2. ADD – sudeda du viršutinius steko elementus, sumažina steko rodyklę SP vienetu ir padeda rezultatą į steko viršūnę.
   ST [SP – 1] = ST [SP – 1] + ST [SP]; SP--;
3. SUxy – atima iš registro reikšmę, esančią adresu xy;
4. SUB – atima steko viršūnėje esantį elementą iš antro nuo viršaus steko elemento, steko rodyklę SP sumažina vienetu bei rezultatą priskiria steko viršūnei.
   ST [SP – 1] = ST [SP – 1] – ST [SP] ; SP--;

5. MUL – sudaugina du viršutinius steko elementus, sumažina steko rodyklę SP vienetu ir padeda rezultatą į steko viršūnę.
   ST [SP – 1] = ST [SP – 1] * ST [SP]; SP--;
6. DIV – padalina antrą nuo viršaus steko elementą iš viršūnėje esančiojo, sumažina SP vienetu ir padeda rezultatą į steko viršūnę. ST [SP – 1] = ST [SP – 1] / ST [SP]; SP--;

3.3.2 Palyginimo komanda

1. CMxy – palygina registre R esantį žodį su xy adrese esančiu žodžiu;
2. CMP –palygina registre R esantį žodį su steko viršūnėje esančiu žodžiu ir pagal palyginimo rezultatą formuoja registro C reikšmę: 1 – jei R = ST [SP], kitu atveju – 0.

3.3.3 Darbo su duomenimis komandos

1. LRxy – į registrą R užkrauna žodį iš atminties nurodytu adresu xy.
2. SRxy – registre R esantį žodį patalpina adresu xy.

3.3.4 Steko operacijos

1. PUSH – steko viršūnė SP padidinama vienetu ir į ja patalpinamas registre R esantis žodis.
   SP = SP + 1; ST [SP] = R;
2. POP – steko viršūnėje esantis žodis talpinamas į registrą R ir SP sumažinama vienetu.
   R = ST [SP]; SP--;

3.3.5 Valdymo komandos

1. JMxy – nesąlyginio valdymo perdavimo komanda. Ji reiškia, kad valdymas turi būti perduotas nurodytu adresu xy.
   IC = 10*x + y;
2. JExy – valdymas perduodamas adresu xy, jei C = 1;
3. JNxy – valdymas perduodamas adresu xy, jei C = 0;
4. HALT – programos sustojimo komanda.

3.3.6 Įvedimo/Išvedimo komandos

1. GDxy. Iš įvedimo srauto perskaito 1 žodį ir jį įrašo į virtualios mašinos atmintį adresu xy;
2. PDxy Išsiunčia išvedimui 1 žodį iš atminties ląstelės adresu xy.

3.4 Atmintis

3.4.1 Realios mašinos atmintis

Reali mašina turi dvejų rūšių atmintis: vidinę ir išorinę.

Vidinė atmintis dalinama į dvi dalis: vartotojo ir supervizorinę. Vartotojo atmintis – 1000 žodžių (100 blokų). Joje laikomos virtualių mašinų atmintys ir puslapių lentelės. Supervizorinė atmintis skirta laikyti sisteminiams procesams, kintamiesiems, resursams. Žemiau esančioje schemoje, K nurodo paskutinio atminties bloko numerį skirtą supervizoriniai atminčiai. Šis skaičius yra nežinomas, nes supervizorinė atmintis nebus realizuota.

Išorinė atmintis realizuota failu kietajame diske. Ji sudaryta iš 1000 takelių (1000 * 10 = 10000 žodžių). Failų sistema sudaryta iš konkrečių failų duomenų bei failų lentelės. Failų lentelėje yra saugomi: failo vardas (iki 8 simbolių), failo pradžios ir pabaigos adresai.

Kuriant naują failą ieškoma pirmo tuščių takelių skaičiaus, lygaus arba didesnio už reikiamą failui įrašyti. Jei rasti pavyksta, informacija apie naują failą patalpinama failų lentelėje, o jo duomenys įrašomi į failų sistemą. Visas darbas su išorine atmintimi vyksta per kanalų įrenginį.

000	001	002	003	004		996	997	998	999	Supervizorinė atmintis	K-3	K-2	K-1	K

Figūra 6: Vidinė atmintis

3.4.2 Virtualios mašinos atmintis

Virtualios mašinos atminčiai skiriama 10 takelių. Į šią atmintį turi būti talpinama užduoties programa ir stekas. Virtualios atminties pavyzdys:

\ Žodis Takelis	0	1	2	3	4	5	6	7	8	9
0										
1	1234									
2	LR10	PUSH	PUSH	ADD	POP	HALT				
3										
4										
5										
6										
7										
8										
9										

Figūra 7: Virtualios atminties pavyzdys

3.5 Puslapiavimo mechanizmas

Virtuali mašina operuoja virtualiais adresais, todėl jai reikalingas PTR (puslapių lentelės registras), kurio pagalba virtuali mašina operuoja realios atminties duomenimis.

Išskiriant atmintį virtualiai mašinai, būna nustatoma PTR reikšmė, kuri rodo į atminties takelį, kuriame yra 10 realios atminties takelių numerių. Šis takelis vadinamas puslapių lentele.

Realaus takelio numeris	9	10	80	4	6	7	33	1	26	87	23

Figūra 8: Puslapių lentelės pavyzdys

Šia lentelę reikia interpretuoti, kaip sunumeruotą nuo 0 iki 9, kur kiekvienas stulpelis rodo realios mašinos takelių numerius.

PTR registras susideda iš dviejų reikšmių p_1 ir p_2. Puslapių lentelės adresą nurodo $10*p_1+p_2$ reikšmė. Pasinaudodami PTR reikšme galime apskaičiuoti realų adresą – $10*[10*(10*p_1 + p_2) + x_1] + x_2$.

Dabar pateiksime pavyzdį, kaip virtuali mašina gauna reikšmę iš realios atminties, pasinaudodama PTR registru ir virtualiu adresu. Tarkime PTR registro reikšmė yra 45, o pati lentelė atitinka aukščiau esančią figūrą (žr. Figūra 8). Sakykime, kad virtuali mašina vykdo komandą LR10. Ji nori pasiimti reikšmę iš virtualios atminties 1 takelio 0 lauko. Žingsniai, kuriuos reikia atlikti, norint gauti realią reikšmę:

1) Gauname puslapių lentelės adresą remdamiesi PTR registro reikšme. Šiuo atveju – $10*4+5 = 45$.
2) Gauname realaus adreso takelio numerį: $45*10+x_1 = 45*10+0 = 450$ žodis, kuris saugo mus dominančio takelio numerį. Šiuo atveju ta reikšmė – 10.
3) Apskaičiuojam realų žodžio adresą – $10*10+x_2 = 100$.

3.6 Pertraukimų mechanizmas

Pertraukimas – tai signalas apie pasikeitusią mašinos būseną. Pertraukimas nenutraukia sistemos darbo – jis turi būti aptiktas ir atpažintas bei iškviečiama jį apdorojanti paprogramė.

Pertraukimai gali būti aptikti tik vartotojo režime. Kiekvieną kartą įvykdžius komandą, kviečiama komanda test(), kuri ir aptinka pertraukimus.

Pertraukimai skirstomi į tris grupes: programiniai, supervizoriniai ir taimerio.

Programiniai pertraukimai kyla, virtualiai mašinai mėginant įvykdyti kokį nors neleistiną veiksmą. Programiniai pertraukimai (PI reikšmė):

- 1 – klaidingas adresas;
- 2 – neteisingas operacijos kodas,;
- 3 – perpildymas (overflow).

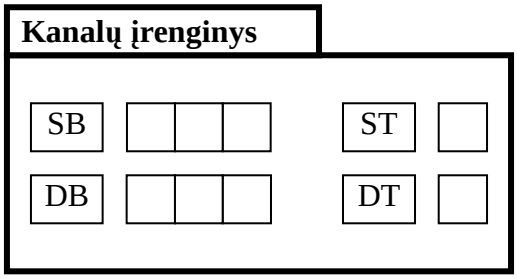
Supervizoriniai pertraukimai kyla, kai virtuali mašina mėgina įvykdyti komandą, kuri gali vykti tik supervizoriniame darbo režime arba, kai vartotojas nori įvykdyti naują komandą. Supervizoriaus režime centrinio procesoriaus darbo pertraukti negalima. Supervizoriniai pertraukimai (SI reikšmė):

- 1 – komanda GD;
- 2 – komanda PD;
- 3 – komanda HALT;
- 4 – vartotojo komanda.

3.7 Taimerio mechanizmas

Taimerio mechanizmas atsakingas už procesų išlygiagretinimą. Ta pati užduotis negali būti vykdoma daugiau nei N taktų. Iš pradžiu taimerio registrui (TMR) priskiriama pradinė reikšmė, kuri yra mažinama vienetu po kiekvienos įvykdytos komandos. Kai TMR pasidaro lygi 0, registro IR laukas TI pasikeičia į 1, ir pertraukimų mechanizmas aptinka taimerio pertraukimą. Tuomet procesų planuotojas apsprendžia kam turi būti perduotas procesoriaus resursas.

3.8 Duomenų perdavimo kanalai



Figūra 9: Kanalų įrenginys

Kanalų įrenginys vykdo apsikeitimą duomenimis priklausomai nuo nustatytų registrų reikšmių. Kanalų įrenginio registrai:

- SB – takelio, iš kurio kopijuosime, numeris;
- DB – takelio, į kurį kopijuosime, numeris;
- ST – objekto, iš kurio kopijuosime, numeris;
- DT – objekto, į kurį kopijuosime, numeris;

Objektų numeriai:

1. Vartotojo atmintis;
2. Supervizorinė atmintis;

3. Išorinė atmintis;
4. Įvedimo/išvedimo srautas;

Procesai norėdami naudotis kanalų įrenginiu, pirmiausia turi gauti „kanalų įrenginio resursą“ (plačiau skyriuje „Resursai“).

3.9 Užduoties formatas

Užduotis yra kodas, kurį vykdys virtuali mašina. Užduotys laikomos išorinės atminties failuose. Taip pat, galima užkrauti bet kokį failą esantį kompiuteryje, nurodant pilną arba reliatyvų kelią iki jo (žr. 2 priedas. Operacinės sistemos komandos).

Užduotis susideda iš 100 žodžių. Užduoties pavadinimą atitinka failo vardas, kuriame ji saugoma. Užduoties struktūra: duomenų segmento pradžią nurodo simboliai &&DS. Kodo segmentas žymimas &&CS. Duomenų segmentui skiriami du pirmieji virtualios mašinos takeliai. Duomenys įrašomi į atmintį tokia tvarka, kokia yra išvardinti, t.y. pirmasis nurodytas duomuo įrašomas į 00 adresą, sekantis į 01 ir t.t.

Pavyzdinė užduotis:

$$DS
0004
$$CS
LR00
PUSH
PUSH
MUL0[3]
POP0
SR80
PT80
HALT
$$$$

Šios užduoties rezultatas yra adrese 00 esančios reikšmės kvadratas, išvestas į išvedimo įrenginį.

[3] Kadangi užduoties formate kiekviena komanda turi susidėti iš 4 baitų, tai prie komandų iš 3jų simbolių pridedamas ketvirtas simbolis, kuris gali būti bet koks.

4 Operacinės sistemos modelis

Absoliuti dauguma šiuolaikinių operacinių sistemų yra multiprograminės, t.y. vienu metu vykdo keletą procesų. Iš tiesų, vienu metu yra vykdomas tik vienas procesas, tačiau persijungimas tarp procesų užtrunka vos kelias mili sekundes, todėl naudotojui sukuriamas lygiagretumo pojūtis.

4.1 Procesai

Programa – tai baitų rinkinys, kurį gali vykdyti procesorius, tuo tarpu procesas – tai vykdoma programa, įskaitant esamas registrų ir kintamųjų reikšmes. Kitaip tariant, procesas – tai tam tikroje vykdymo stadijoje esanti programa. Visa operacinės sistemos vykdoma programinė įranga susideda iš atskirų procesų.

Procesai gali būti dviejų rūšių: sisteminiai ir vartotojiški. Sisteminių procesų paskirtis aptarnauti vartotojiškus procesus, tuo tarpu vartotojiški procesai skirti naudotojo poreikių tenkinimui.

Procesai gali būti sukuriami šiais pagrindiniais atvejais:

- Sistemos inicializavimo metu;
- Paleistam procesui pareikalavus;
- Naudotojui pareikalavus (pvz., įvedus užduoties failo vardą).

Techniškai naują procesą gali sukurti tik kitas procesas. Operacinės sistemos paleidimo metu yra sukuriamas vienas ar daugiau procesų, kurie vėliau kuria visus kitus procesus. Procesų manipuliacijai skirti primityvai aprašyti „Procesų primityvai“ skyriuje.

4.2 Proceso būsenos

Einamuoju momentu procesas yra tam tikroje būsenoje. Būsenas keisti galima pasinaudojant primityvais, todėl perėjimas iš vienos būsenos į kitą pateikiamas lentelėje Figūra 12, esančioje „Procesų primityvai“ skyriuje. Procesai gali įgyti vieną iš šių būsenų.

1. Vykdomas – tai procesas, kuris einamuoju momentu yra vykdomas, t.y. jam priklauso procesoriaus resursas;
2. Pasiruošęs – tai procesas, kuris turi visus jam reikalingus resursus, išskyrus procesorių;
3. Blokuotas – tai procesas, kuris pareikalavo kažkokio resurso ir šis jam dar nebuvo suteiktas.
4. Blokuotas sustabdytas – procesas yra ne tik užblokuotas, bet dar ir sustabdytas.
5. Pasiruošęs sustabdytas – šioje būsenoje esantys procesai turi visus reikalingus resursus, tačiau negali būti vykdomi iki kol jiems aktyvuoti bus iškviestas atitinkamas primityvas.

4.3 Proceso deskriptorius

Procesą apibūdina tam tikra struktūra, kuri vadinama proceso deskriptoriumi. Čia pateikiamas java programavimo kalba parašytas kuriamos sistemos proceso aprašas, kartu pateikiant ir su juo susijusias struktūras:

```
class Procesas {
    AbstractProcess process;
    Busena state;
    int id;
    int priority;
    int defaultPriority;
    String name;
    Procesorius registrai;
    int pId;
    LinkedList<Integer> childProcesses;
    LinkedList<Elementas> ownedResourceElements;
    LinkedList<Integer> createdResources;
}
enum Busena { BLOCKED, READY, ACTIVE, BLOCKED_STOPPED, READY_STOP };
class Procesorius {

    char[] PTR = {'0', '0'};
    char[] R = {'0', '0', '0', '0'};
    char[] IC = {'2', '0'};
    char MODE = 'S';
    char[] IR = {'0', '0', '0'};
    char[] TMR = {'0', '9'};
    char C = '0';
    char[] SP = {'7', '9'};
    int ioAddr;
}
abstract class AbstractProcess {
    Procesas itself;
    int state;
    abstract void doit();
}
```

Figūra 10: Proceso ir su juo susijusių klasių aprašai

Klasių kintamųjų pavadinimai vienareikšmiškai apibrėžia jų paskirtį. Būtina paminėti, kad klasės Procesorius ir AbstractProcess turi nuoroda viena į kitą ir abi atstovauja vieną ir tą patį procesą. Tokį pasirinkimą lėmė tai, kad branduolio primityvams patogiau naudoti procesą kaip klasės „Procesas“ atstovą, tuo tarpu vykdyti procesą patogiau, kai galime kviesti jo vykdomąją funkciją „doIt()“. Be to, kintamasis state abiejose klasėse išreiškia skirtingas būsenas: vienu atveju tai tipinė proceso būsena (žr. skyrių Proceso būsenos), kitu atveju nurodo, kokį žingsnį turėtų vykdyti procesas kitą kartą jį aktyvavus.

4.4 Resursai

Resursai yra tai, dėl ko varžosi procesai. Jei procesui trūksta kažkokio resurso, tuomet jis blokuojasi iki tol, kol tas resursas bus sukurtas arba atlaisvintas. Patį procesorių taip pat galima įsivaizduoti kaip resursą, kuris būtinas kiekvienam procesui.

Resursus galima suskaidyti į dvi grupes:

- Statiniai resursai. Tai resursai, kurie sukuriami operacinės sistemos paleidimo metu, pavyzdžiui, procesorius ir atmintis.
- Dinaminiai resursai. Šie resursai būna sukuriami ir naikinami vykstant operacinės sistemai darbui. Tai gali būti vieno proceso siunčiama užduotis (pranešimas) kitam procesui. Pavyzdys galėtų būt „MOS pabaigos resursas“, kurio laukia užsiblokavęs „Root“ procesas.

Patys resursai naudojami per deskriptorius, kurie saugo visą informaciją apie resursą, kuri būtina norint juo naudotis. Dažniausiai būna saugoma tokia informacija: resurso tipas, elementų kiekis, nuoroda į tikrąjį resursą, nuoroda į resursą sukūrusį procesą ir nuoroda į sąrašą, kuriame yra to resurso laukiantys procesai.

Manipuliacija per resursus vyksta per resurso primityvus (detaliau skyriuje „Resurso primityvai“).

4.5 Resurso deskriptorius

Šiame skyriuje pateikiamas java programavimo kalba parašytas resurso aprašas, naudojamas kuriamoje modelinėje sistemoje.

```
class Resursas {
    int amount;
    String name;
    int id;
    int pId;
    boolean reusable;
    LinkedList<Elementas> availableElements;
    LinkedList<LaukiantisProcesas> waitingProcesses;
}
abstract class Elementas {
    String name;
    int target;
    boolean reusable;
}
class LaukiantisProcesas {
    private int id;
    private int amount;
}
```

Figūra 11: Resurso ir su juo susijusių klasių aprašai

Resursas saugo sąrašus nuorodų į savo elementus bei šio resurso laukiančius procesus. Klasė LaukiantisProcesas saugo proceso numerį (id) ir skaičių elementų (amount), kuris reikalingas proceso darbui pratęsti. Resursas turi laisvų elementų skaičiaus kintamąjį tuo pačiu pavadinimu (amount), kuris kaskart sumažinamas, kuomet būna išskiriamas elementas ir padidinamas, kai sukuriamas arba atlaisvinamas.

4.6 Multiprograminės operacinės sistemos branduolio primityvai

4.6.1 Procesų primityvai

4.6.1.1 Kurti procesą

Šiam primityvui perduodami pradiniai duomenys: nuoroda į proceso tėvą, jo pradinė būsena, prioritetas, perduodamų elementų sąrašas ir išorinis vardas. Pačio primityvo viduje vyksta proceso kuriamasis darbas. Jis užregistruojamas bendrame procesų sąraše, tėvo-sūnų sąraše, skaičiuojamas

vidinis identifikacijos numeris, sukuriamas jo vaikų procesų sąrašas (tuščias), sukurtų resursų sąrašas.

4.6.1.2 Naikinti procesą

Procesas yra pašalinamas iš visų procesų sąrašų, prieš tai sunaikinant proceso sukurtus resursus ir jo vaikus.

4.6.1.3 Stabdyti/aktyvuoti procesą

Šie primityvai keičia einamąją proceso būseną. Daugiau apie būsenas skyriuje „Proceso būsenos“.

Pradinė būsena	**Veiksmas**	**Galutinė būsena**
Blokuota	Stabdyti	Blokuota sustabdyta
Pasiruošusi	Stabdyti	Pasiruošusi sustabdyta
Vykdoma	Stabdyti	Pasiruošusi sustabdyta
Blokuota sustabdyta	Aktyvuoti	Blokuota
Pasiruošusi sustabdyta	Aktyvuoti	Pasiruošusi

Figūra 12: Procesų būsenos

4.6.2 Procesų planuotojas

Pasirinktas prioritetais besiremiantis procesų planuotojo modelis. Žemiausias – 1, didžiausias – 100. Siekiant sistemos efektyvumo, sisteminiai procesai turės aukštesnį prioritetą lyginant su vartotojiškais procesais. Todėl vartotojo proceso prioritetas gali būti tik mažesnis už 81. Tuo tarpu sisteminiai procesai turės prioritetus nuo 81 iki 100 imtinai. Kiekvieną kartą vartotojo procesui pakeitus būseną iš vykdomos į kurią nors kitą, jo prioritetas mažinamas vienetu. Jei procesas pakeitė būseną dėl taimerio pertraukimo, tuomet kitą kartą jį vykdant TMR registrui bus priskirta 10 vienetų didesnė reikšmė. Ši reikšmė negali viršyti 40. Tokiu būdu procesoriaus resursą ilgesnį laiką turės tie procesai, kurių darbo trukmė ilga. Toks procesų vykdymo modelis leis išvengti nebūtino procesų lygiagretinio kartu išlaikant sistemos interaktyvumą.

Sisteminių procesų prioritetas niekuomet nesikeičia.

Procesų planuotojas iškviečiamas aptikus taimerio pertraukimą, sukūrus naują procesą, procesui baigus darbą arba jam pakeitus būseną iš vykdomos į blokuotą. Kitas vykdomas procesas parenkamas pagal didžiausią prioritetą iš pasiruošusių procesų sąrašo.

4.6.3 Resurso primityvai

4.6.3.1 Kurti resursą

Parametrai perduodami kuriant resursą: nuoroda į proceso kūrėją, resurso išorinis vardas. Resursas kūrimo metu pridedamas prie bendro resursų sąrašo, pridedamas prie tėvo sukurtų resursų sąrašo, jam priskiriamas unikalus vidinis vardas, sukuriamas resurso elementų sąrašas ir sukuriamas laukiančių procesų sąrašas.

4.6.3.2 Naikinti resursą

Resurso deskriptorius šalinamas iš bendro resursų ir jo tėvo sukurtų resursų sąrašų, naikinamas jo elementų sąrašas, atblokuojami procesai laukiantys šio resurso, vėliau naikinamas pats deskriptorius.

4.6.3.3 Prašyti resurso

Procesas iškvietęs šį primityvą yra blokuojamas ir įtraukiamas į to resurso laukiančių procesų sąrašą.

4.6.3.4 Atlaisvinti resursą

Resurso elementas yra perduodamas šiam primityvui kaip parametras ir yra pridedamas prie resurso elementų sąrašo.

4.6.4 Resursų paskirstytojas

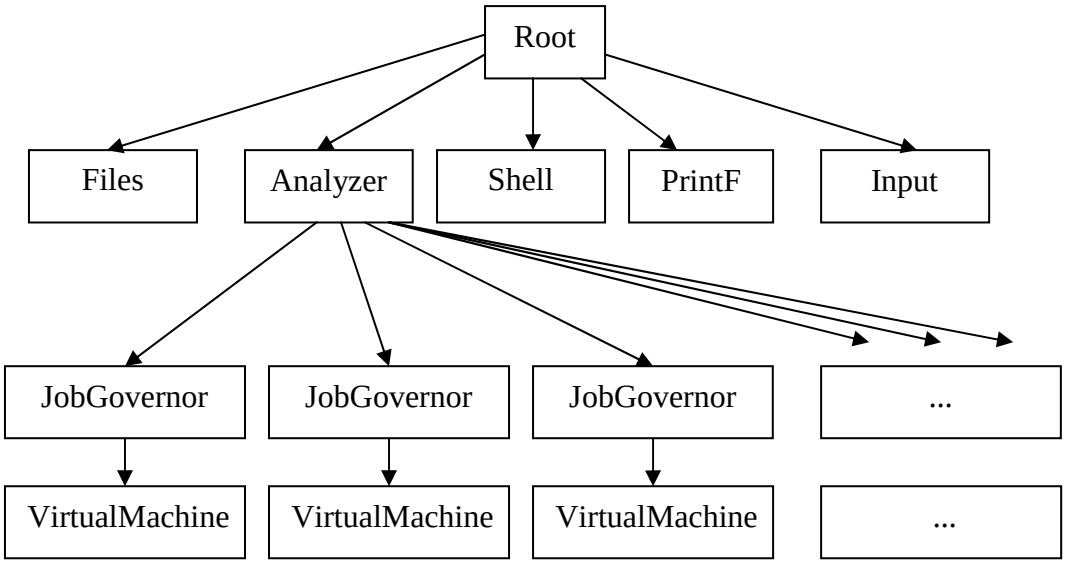
Resursų paskirstytojo paskirtis – suteikti paprašytą resurso elementų kiekį procesui. Jis, kaip ir procesų planuotojas, paremtas prioritetais. Resursų paskirstytojas iškviečiamas kuriam nors procesui prašant resursų, atlaisvinant arba sukuriant resursą. Atsilaisvinus resursui, jis perduodamas šio resurso laukiančiam procesui su didžiausiu prioritetu.

4.7 OS sisteminiai procesai

Šiame skyriuje aprašome atskirus operacinės sistemos procesus. Jie būdingi tik mūsų kuriamai sistemai ir to nereikia priimti kaip taisyklės.

4.7.1 Procesų medis

Figūroje „OS sisteminių procesų medis“ pavaizduota visų operacinėje sistemoje esančių procesų sąveika. Rodyklės rodo tėvinio ir vaikinio procesų sąveiką. Pvz., Root procesas sukuria Analyzer procesą, kuris savo ruožtu gali sukurti daug JobGovernor procesų.

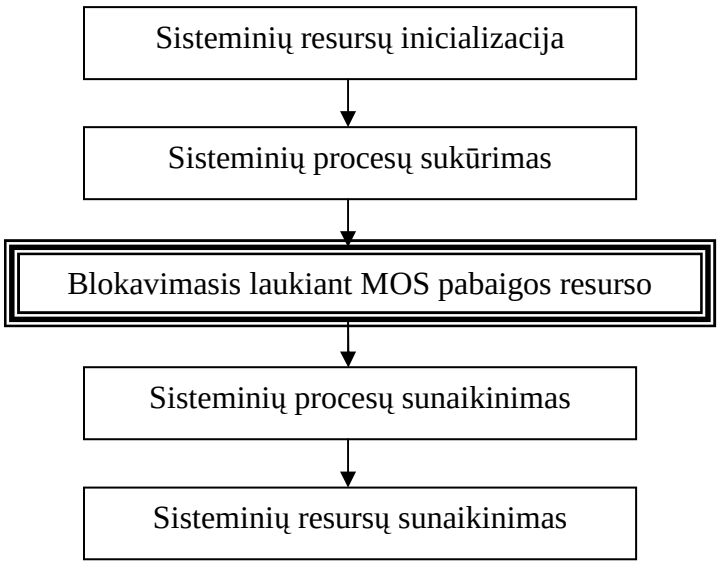


Figūra 13: OS sisteminių procesų medis

Dabar apžvelgsime visus procesus atskirai

4.7.2 Root

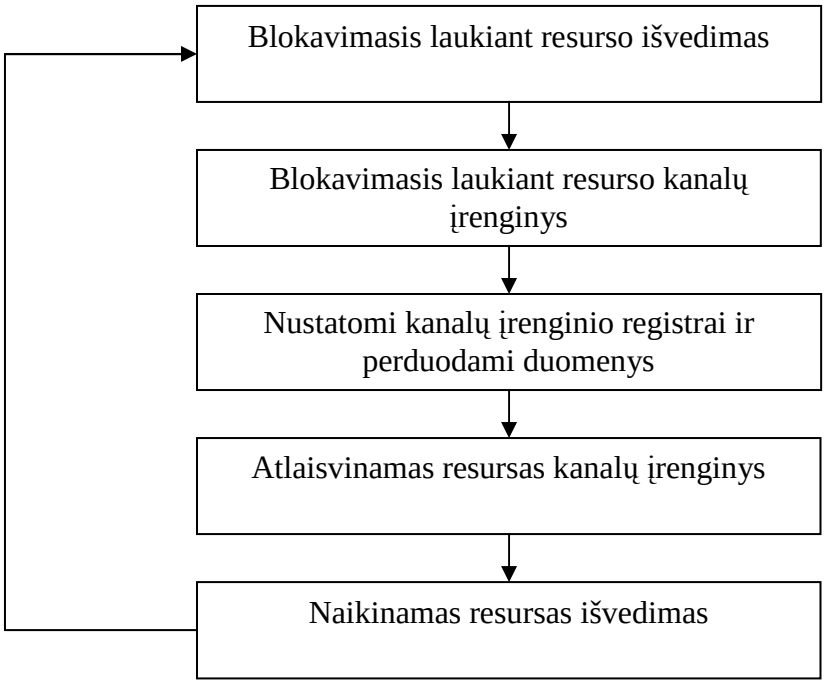
Root procesas yra šakninis procesas, kuris sukuria kitus sisteminius procesus bei sistemos resursus. Sistemos darbo pabaigoje juos sunaikina.



Figūra 14: Root proceso vyksmo logika

4.7.3 PrintF

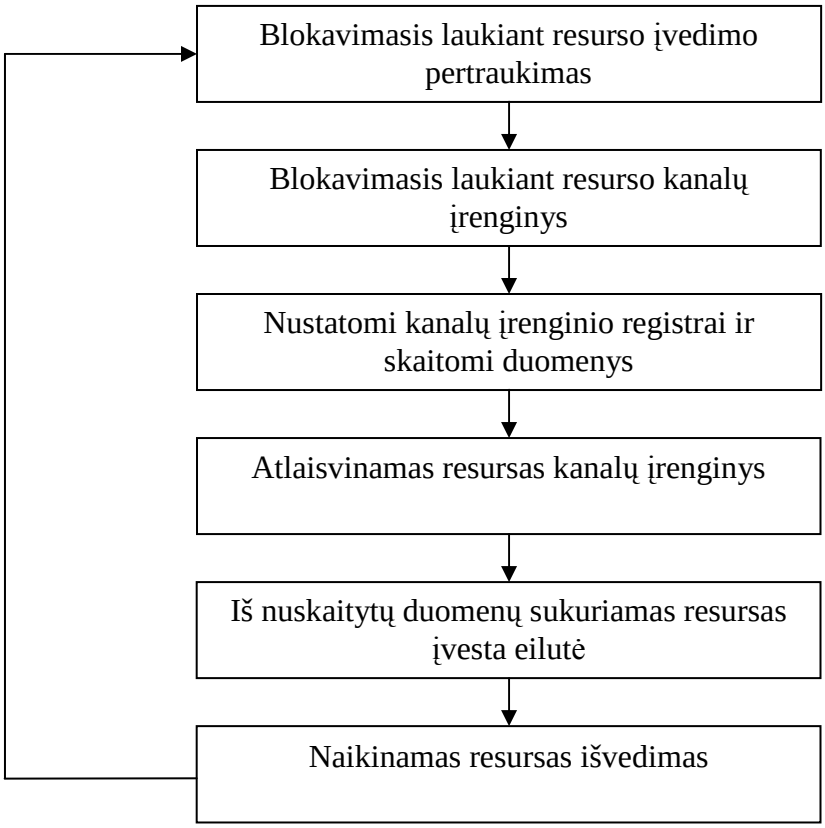
Procesas skirtas duomenų išvedimui pasinaudojant kanalų įrenginiu.



Figūra 15: PrintF proceso vyksmo logika

4.7.4 Input

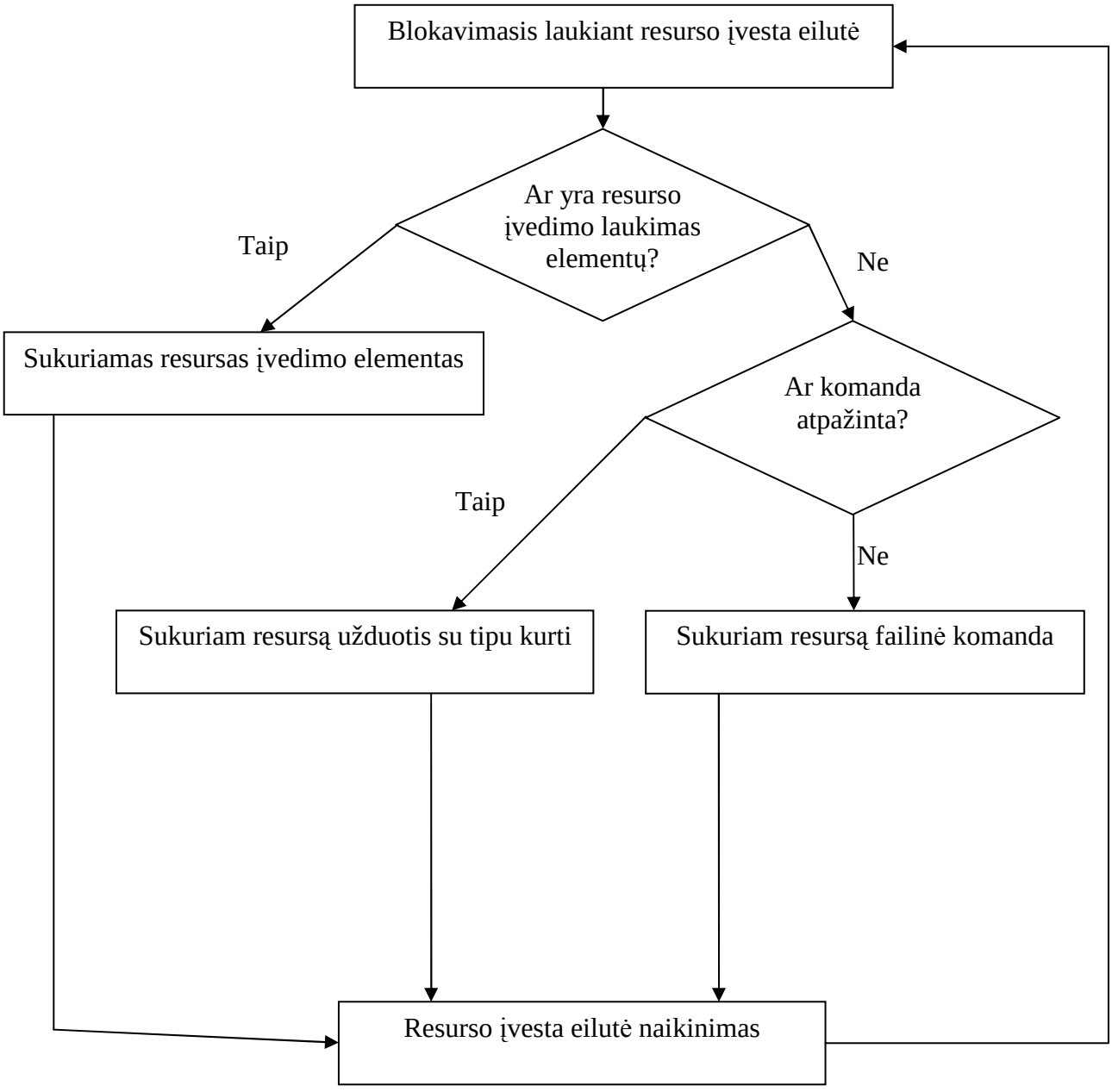
Šis procesas užsiblokavęs laukia resurso įvedimo pertraukimas, vėliau nuskaito duomenis iš įvedimo įrenginio ir sukuria resursą įvesta eilutė.



Figūra 16: Input proceso vyksmo logika

4.7.5 Shell

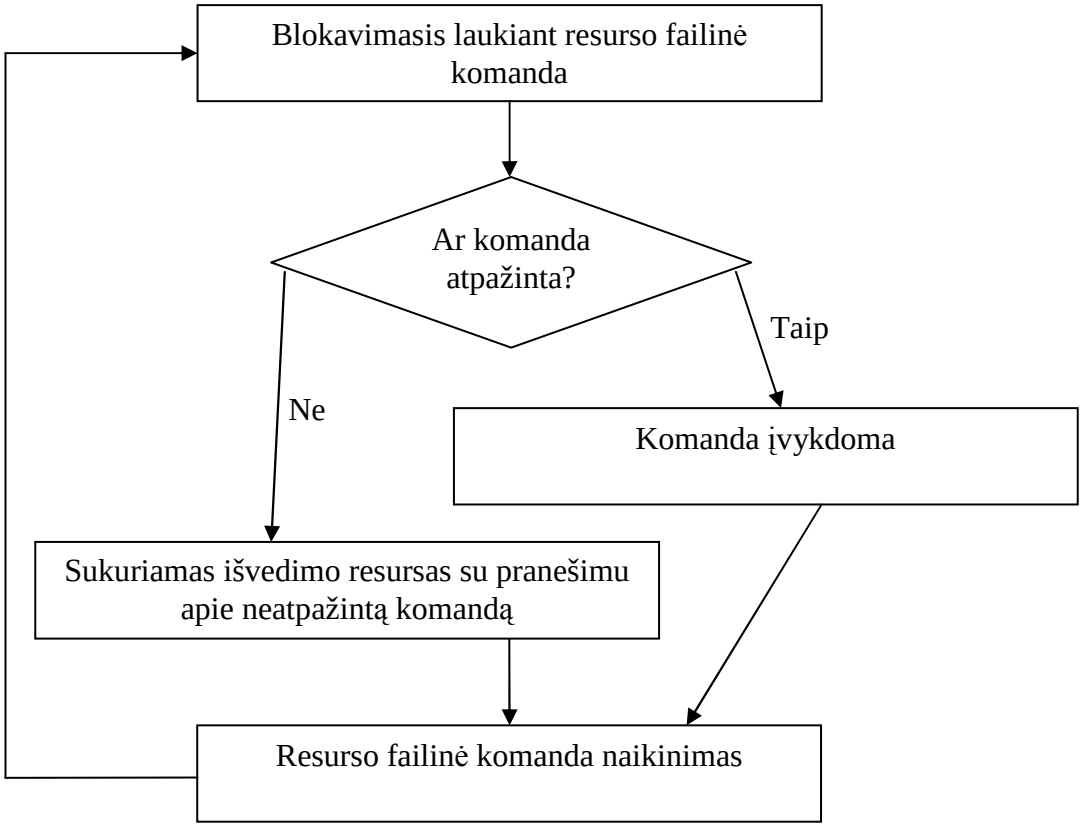
Procesas sukuria resursą „užduotis“, kuomet iš naudotojo gauna komandą, su nurodytu failo vardu, kuriame yra patalpinta užduotis. Vėliau šią užduotį apdoroja procesas Analyzer



Figūra 17: Proceso Shell vyksmo logika

4.7.6 Files

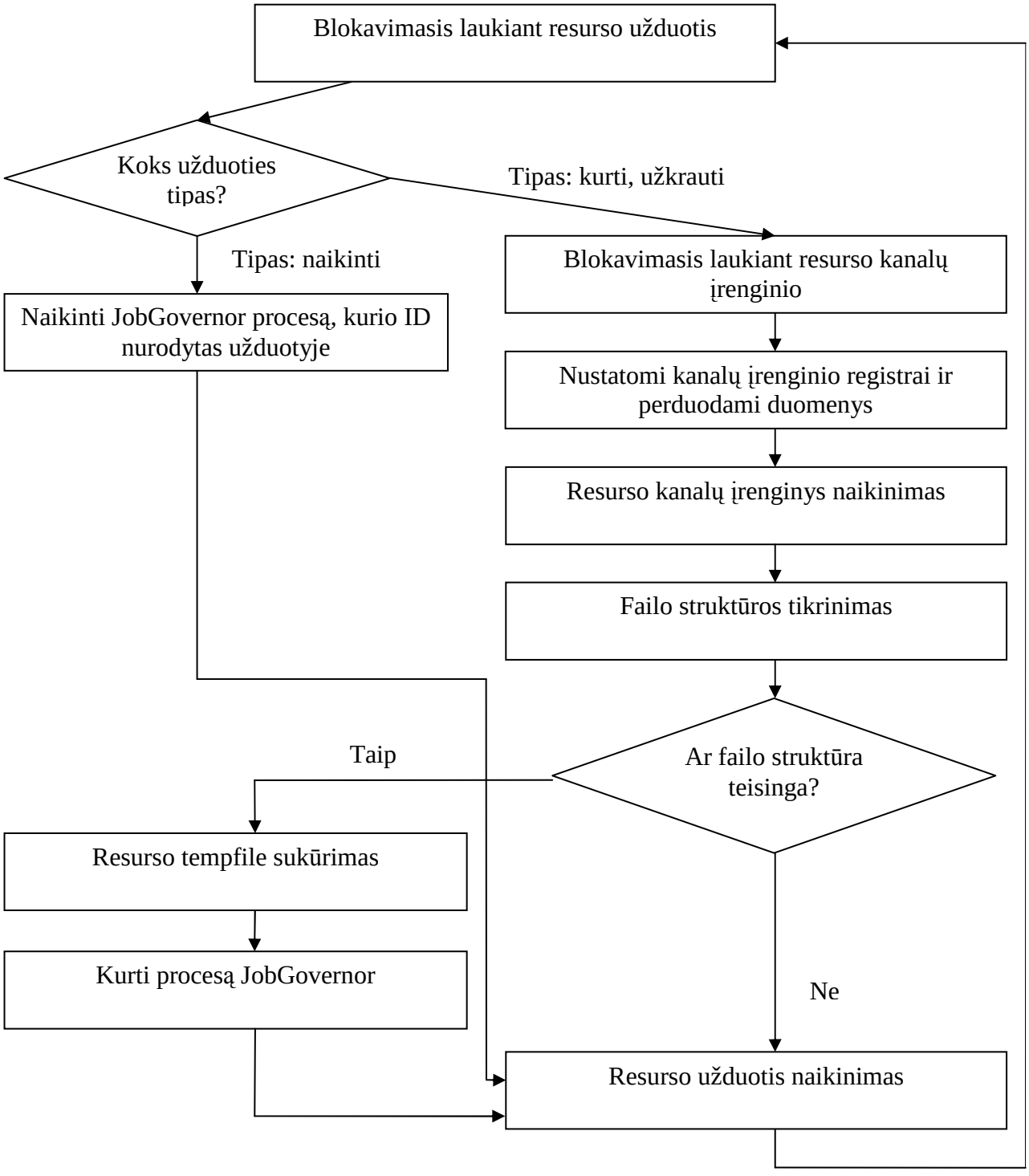
Šis procesas skirtas failų sistemos komandoms apdoroti.



Figūra 18: Proceso Files vyksmo logika

4.7.7 Analyzer

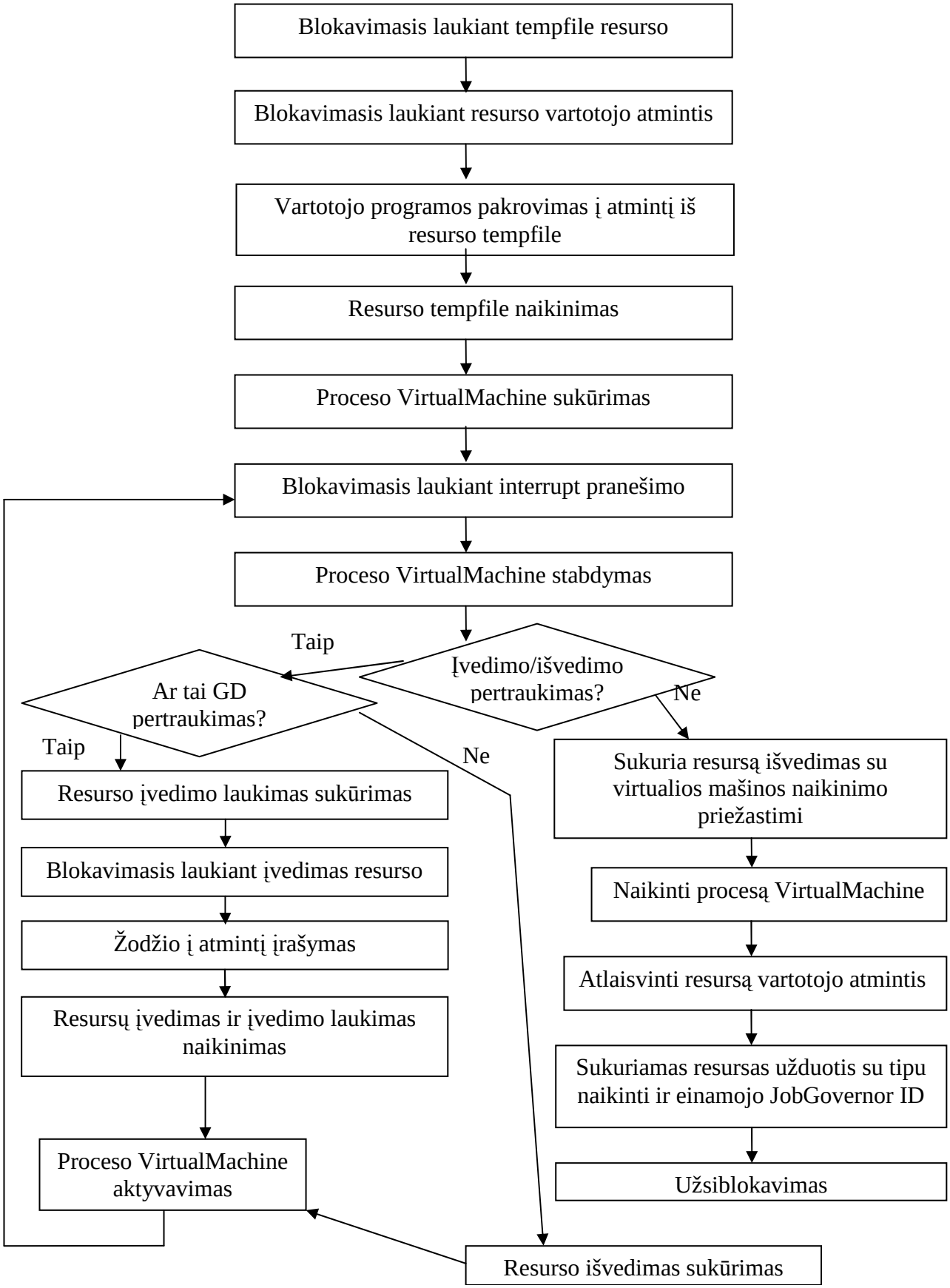
Analyzer procesas apdoroja resursą „užduotis“. Priklausomai nuo užduoties tipo, jis gali sukurti arba sunaikinti procesą JobGovernor. Taip pat jis patikrina naudotojo užduoties kodo korektiškumą.



Figūra 19: Analyzer proceso vyksmo logika

4.7.8 JobGovernor

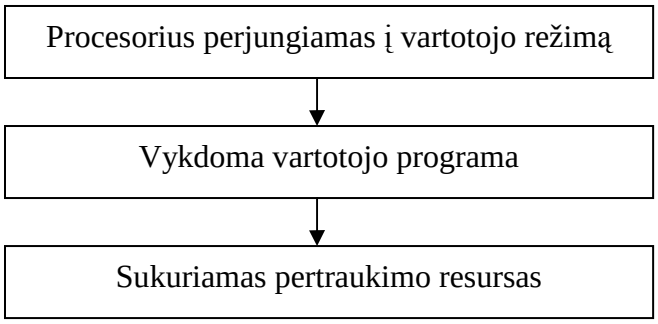
Šis procesas apdoroja proceso Analyzer sukurtą resursą “tempfile” ir sukuria procesą VirtualMachine. Taip pat yra tarpininkas tarp VirtualMachine ir pertraukimų mechanizmo.



Figūra 20: JobGovernor proceso vyksmo logika

4.7.9 VirtualMachine

Tai procesas vykdantis naudotojo užduoties kodą. Šis procesas elgiasi taip, tarsi jis būtų vienintelis procesas visoje sistemoje.



Figūra 21: VirtualMachine proceso vyksmo logika

5 Multiprograminės operacinės sistemos projekto realizacija

Ankstesniuose skyriuose buvo apibrėžta reali mašina ir operacinės sistemos modelis. Šiame skyriuje dėmesys skiriamas šio modelio realizacijai (kaip atrodo realizuota sistema galima pamatyti pateiktuose prieduose).

Projekto įgyvendinimui buvo pasirinkta java programavimo kalba. Sistema pagal atliekamų funkcijų prasmę suskirstyta į keturis paketus (package). Taip pat, kiekvieną atskirą sistemos vienetą atitinka tam skirta klasė, pavyzdžiui, RealiMašina klasė atitinka visą kompiuterio aparatūrinę įrangą.

Dabar detaliau apžvelgsime kiekvieną paketą.

5.1 Paketas „kompiuteris“

Paketas, kurio klasės imituoja tikro kompiuterio architektūrą. Jis susideda iš šių svarbiausių klasių:

- OperacinėsSistemosLangas – pagrindinis sistemos langas, kuriame pateikiama visa informacija apie operacinės sistemos veikimą bei jos išvedimas, taip pat, šiame lange galima įvesti komandas;
- VMLangas – langas, kuriame pateikiama aktyvios virtualios mašinos būsena;
- IšvedimoĮrenginys – ši klasė atsakinga už duomenų atvaizdavimą ekrane;
- KanalųĮrenginys – klasė skirta duomenų apsikeitimui tarp kompiuterio dalių;
- Procesorius – susideda iš registrų ir procesoriaus atliekamų komandų;
- VartotojoAtmintis – tai klasė kuri susideda iš masyvo masyvų, t.y., vieną masyvo elementą atitinka takelis, o vieną takelio elementą atitinka vienas žodis.
- IšorinėAtmintis – klasė, per kurią bendraujama su išorine atmintimi;
- ŽurnalizavimoĮrenginys – klasė, atsakinga už išvedamus pranešimus apie operacinės sistemos atliekamus veiksmus;
- RealiMašina – apjungia ankščiau išvardintas klases į vieną visumą.

5.2 Paketas „modelineos“

Šiame pakete laikomos klasės, kurios atsakingos už sistemos paleidimą bei funkcionavimą.

- Main – ši klasė skirta paleisti operacinės sistemos gijai;
- Branduolys – vieta, kur saugomi procesų sąrašai bei resursų ir procesų primityvus atitinkančios funkcijos;
- PertraukimųApdorojimas – klasė, kuri kiekvienam pertraukimui apdoroti saugo po atskirą funkciją;

- PuslapiavimoMechanizmas – klasė, apskaičiuojanti realų adresą pagal pateiktą registro reikšmę;
- ProcesųPlanuotojas – klasė, kuri atsakinga už tai, kad aktyviu procesu taptų didžiausią prioritetą bei visus reikalingus resursus turintis procesas;
- ResursųPaskirstytojas – klasė, išskirianti resursus procesams;

5.3 Paketas „procesai“

Visi procesai pavadinti angliškai siekiant, kad jų pavadinimai išskirstų pranešimuose apie įvykius sistemoje. Detalus kiekvieno proceso veikimo logikos aprašas pateikiamas skyriuje „OS sisteminiai procesai“.

5.4 Paketas „resursai“

Resursų organizavimas programoje vyksta taip:

- Visos resursų rūšys sukuriamos procese sistemos darbo pradžioje, procese Root;
- Kiekvienos rūšies resursas yra klasės „Resursas“ atstovas, besiskiriantis tik nurodytu pavadinimu;
- Klasė „Resursas“ savyje saugo Elementas klasės tipo objektų sąrašą;
- Elementas gali būti įvairių rūšių (žiūrėti žemiau esančią lentelę)

Resurso pavadinimas	**Elemento pavadinimas**	**Sukuriantys procesai**	**Naudojantys procesai**
Užduotis	UžduotiesElementas	JobGovernor	Analyzer
Pertraukimas	PertraukimoElementas	PertraukimųApdorojimas[4]	JobGovernor
ĮvedimoPertraukimas	PertraukimoElementas	PertraukimųApdorojimas	Input
MOSpabaiga	EilutėsElementas	Shell	Root
Įšvedimas	IšvedimoElementas	Files, JobGovernor	PrintF
Įvedimas	EilutėsElementas	Shell	JobGovernor
UžduotiesFailas	UžduotiesFailoElementas	Analyzer	JobGovernor
Atmintis	VartotojoAtmintiesElementas	Root	JobGovernor
ĮvestaEilutė	EilutėsElementas	Input	Shell
FailinėKomanda	FailųKomandosElementas	Shell	Files
ĮvedimoLaukimas	EilutėsElementas	JobGovernor	Shell
KanalųĮrenginys	KanalųĮrenginys	Root	Analyzer, Input, PrintF

Lentelė 1: Resursų ir procesų sąryšiai

[4] Ši klasė nėra procesas, tačiau yra atsakinga už pertraukimų apdorojimą, todėl gali sukurti šio resurso tipo elementus.

Išvados

Darbe teorija pateikta kartu su viena iš galimų jos interpretacijų, t.y. sukurtos operacinės sistemos projektu. Be to, studentui palikta daug erdvės įvairioms šio projekto modifikacijoms (žr. 1 priedas. Sistemos modifikavimo galimybės). Tokį išdėstymą lėmė vienas iš darbo tikslų. Buvo siekiama, kad studentas pagal šį darbą galėtų parengti savo projektą ir vėliau jį realizuoti.

Visi tikslai, kurie buvo užsibrėžti darbo pradžioje, pasiekti:

- Atrinkta ir pateikta svarbiausia informacija, reikalinga operacinės sistemos darbo principams suprasti;
- Parengtas operacinės sistemos projektas;
- Sukurta modelinė operacinė sistema.

Sukurta sistema turi plačias savęs stebėjimo galimybes, todėl studentas stebėdamas sistemos veikseną ir būsenų kitimus, turėtų gebėti lengvai perprasti multiprograminių operacinių sistemų veikimo principus. Be to, kartu su šiuo darbu pateikiami išeities tekstai, kurie turėti palengvinti programavimo darbus, sutelkiant dėmesį į esmę, o ne smulkmenas.

Literatūros sąrašas

- G. Šiaulys. *Magistro darbas „Mokomoji operacinė sistema“*. Vilnius, 2003.
- A.S. Tanenbaum, A.S.Woodhull. *Operating Systems Design and Implementation (3rd Editon)*. Prentice Hall, Amsterdamas, 2006, 1080 psl.
- M. Goldweber, „Enhancing the operating systems course using the MPS or chip hardware simulator“. *Consortium for Computing Sciences in Colleges*, Tomas 7, 2000, p. 40 – 43.
- O.P. Sharma, „Enhancing operating system course using a comprehensive project: decades of experience outlined”. *Journal of Computing Sciences in Colleges*, Numeris 22, 2007, p. 206 – 213.

1 priedas. Sistemos modifikavimo galimybės

Dėl pasirinkto informacijos išdėstymo stiliaus, labai paprasta pakeisti vieną ar kitą operacinės sistemos dalį. Čia pateikiama keletas galimų modifikacijų:

- Padaryti mažiau fragmentuotą failų sistemą;
- Pakeisti atminties vienetų dydžius, pavyzdžiui, vienu žodžiu laikyti 8 baitus;
- Sukurti kelis procesorius turinčią sistemą;
- Apibrėžti kitokias procesoriaus komandas bei kitokią struktūrą;
- Pakeisti procesų planuotojo veikseną, pavyzdžiui, vietoj prioritetų naudoti bilietų sistemą;
- Pakeisti procesų vyksmo logiką;

2 priedas. Operacinės sistemos komandos

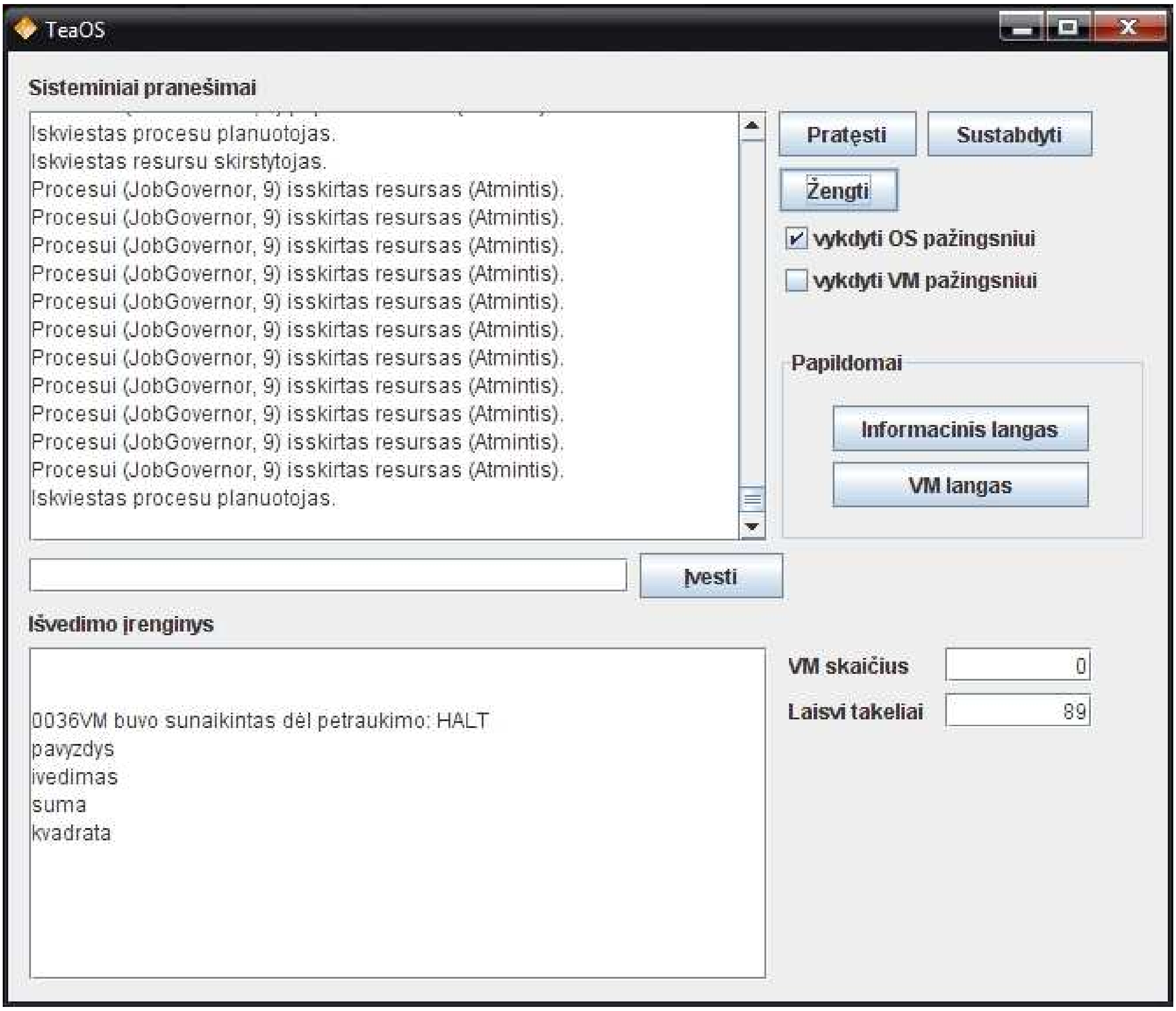
Čia pateikiamas komandų sąrašas, kurias geba įvykdyti sukurta operacinė sistema:

- **show** - išveda visą išorinę atmintį į išvedimo įrenginį. Šis veiksmas labai apkrauna išvedimo įrenginį, todėl gali būti ilgai vykdomas;
- **format** - ištrina visus failus iš išorinės atminties ir paruošia failų sistemą darbui;
- **ls** - išveda išorinės atminties failų sąrašą į išvedimo įrenginį;
- **read file** *failo_vardas* - išveda failo turinį į išvedimo įrenginį;
- **delete file** *failo_vardas* - ištrina failą nurodytu pavadinimu iš išorinės atminties;
- **create file** *failo_vardas* | *failo_turinys* - sukuria naują failą nurodytu pavadinimu bei turiniu;
- **create samples** - sukuria failus su pavyzdinėmis užduotimis;
- **load file** *kelias_iki_failo_ir_failo_vardas* - užkrauna nurodytą failą į atmintį bei iniciazuoja šios užduoties paleidimą. Failo struktūra privalo atitikti aprašytą skyriuje „Užduoties formatas“.
- **failo_vardas** - paleidžia užduoties vykdymą iš išorinės atminties failo. Jei failo struktūra neteisinga, bus išvestas atitinkamas pranešimas.

Taip pat, būtina paminėti, kad komandos išoriniuose failuose gali būti surašytos tiek šalia, tiek atskirose eilutėse, tačiau tarp jų negali būti jokių papildomų simbolių, pvz., tarpų.

3 priedas. Modelinės operacinės sistemos pagrindinis langas

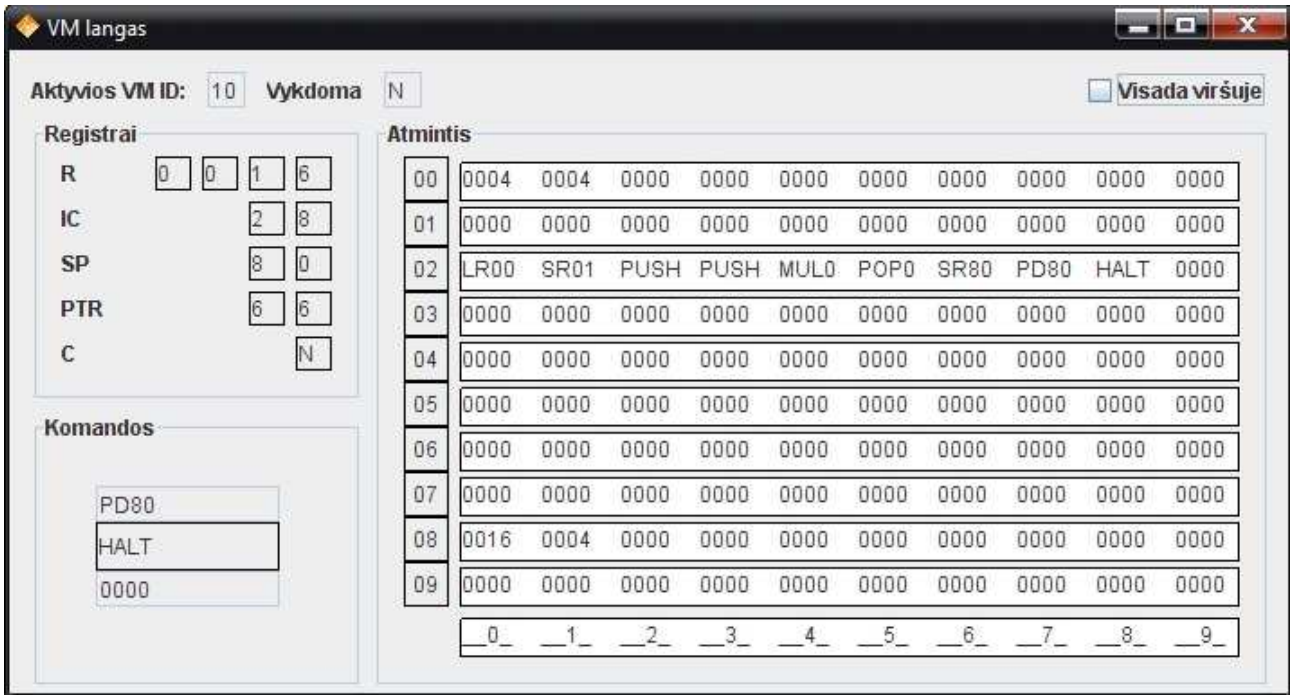
Čia pateikiamas pagrindinis sistemos langas, skirtas informacinių pranešimų išvedimui, komandų įvedimui ir operacinės sistemos išvesčiai.



Figūra 22: Pagrindinis sistemos langas

4 priedas. Virtualios mašinos stebėjimo langas

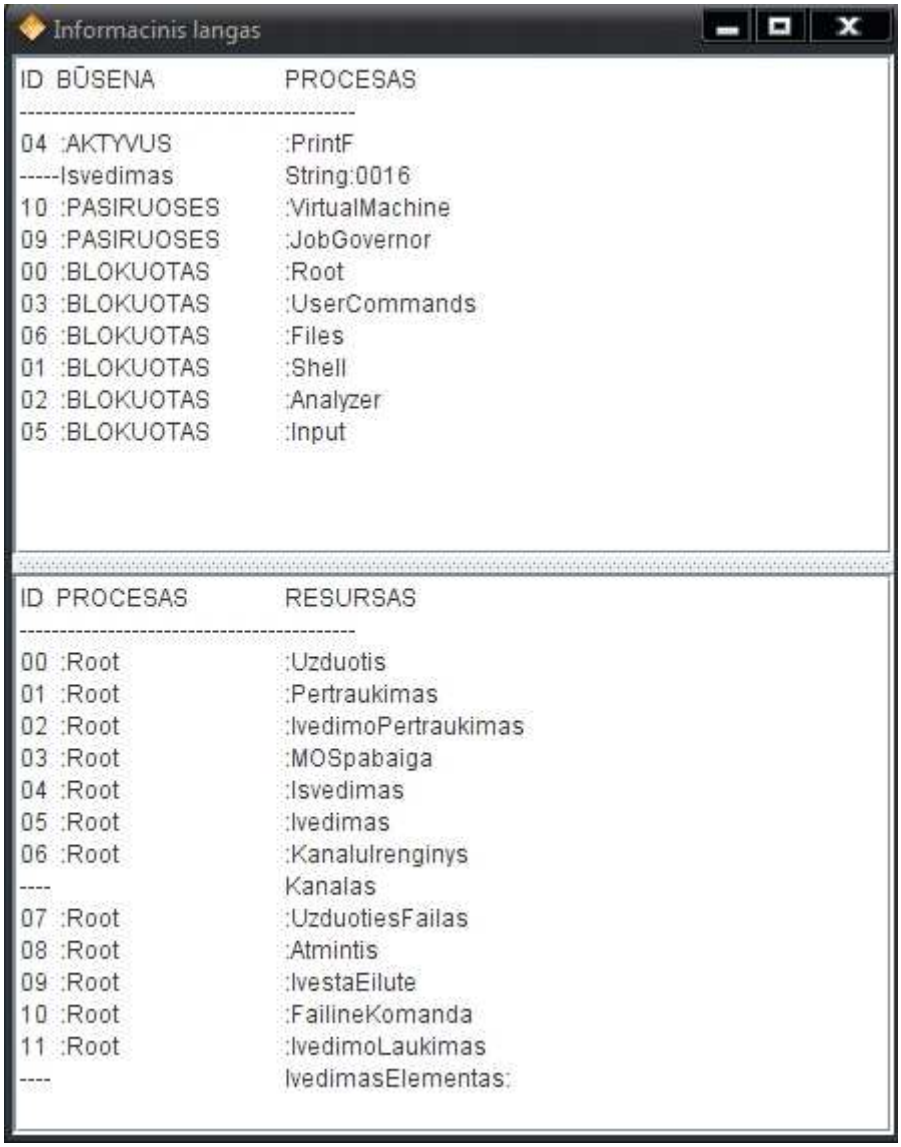
Šį langą pasiekti galima paspaudus mygtuką „VM langas“, esantį pagrindiniame operacinės sistemos lange. Jame pateikiama visa informacija, reikalinga vartotojo užduoties vykdymo proceso stebėjimui.



Figūra 23: Virtualios mašinos stebėjimo langas

5 priedas. Informacinis sistemos langas

Čia pateikiamas procesų ir resursų stebėjimui skirto lango atvaizdas, kurį galima pasiekti paspaudus mygtuką „Informacinis langas“, esantį pagrindiniame operacinės sistemos lange.



Figūra 24: Informacinis sistemos langas