VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ
3 KURSAS

Operacinės sistemos aprašas

Atliko:

Augustinas Jarockis,

Arminas Bražėnas

Vilnius

2025-04

Sąvokos

Naudotojas/vartotojas – operacine sistema besinaudosiantis asmuo.

Operacinė sistema – šiame dokumente aprašyta programa, kurios paskirtis – valdyti pirmame laboratoriniame darbe aprašytos ir antrame laboratoriniame darbe įgyvendintos mašinos resursus.

Procesas – operacinės sistemos vykdoma programa.

Programa – operacijų seka, skirta atlikti konkrečią užduotį.

Sisteminis failas – failas, priklausantis operacinei sistemai.

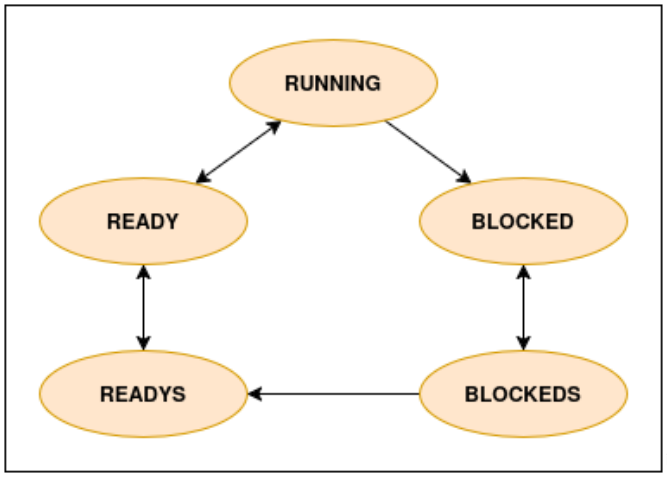
Sisteminis procesas – procesas, priklausantis operacinei sistemai.

Procesai

- Procesas gali būti:
  - Vartotojiškas
    - Skirtas vykdyti vartotojo programą
    - Varžosi tarpusavyje dėl procesoriaus resurso
  - Sisteminis
    - Skirtas aptarnauti vartotojiškus procesus
    - Kuriami paleidžiant sistemą, o naikinami – sustabdant sistemą

Proceso būsenos

- Procesas visada yra vienoje iš šių būsenų:
  - **RUNNING** – procesas yra vykdomas.
  - **READY** – procesas gali būti vykdomas. Vienintelis trūkstamas resursas yra procesorius.
  - **READYS –** procesas galėtų būti vykdomas, tačiau yra pristabdytas. Vienintelis trūkstamas resursas yra procesorius.
  - **BLOCKED** – procesas yra užblokuotas. Procesas laukia kažkokio resurso, išskyrus procesoriaus.
  - **BLOCKEDS** – procesas yra užblokuotas ir pristabdytas. Procesas laukia kažkokio resurso, išskyrus procesoriaus.
- Procesas gali gauti procesorių tik tada, kai jam netrūksta jokio kito resurso *(Ready)*
- Procesas gavęs procesorių tampa vykdomu *(Running)*
- Procesas yra vykdomas, kol sistemoje neįvyksta pertraukimas arba procesas neprašo kokio nors resurso



Planuotojas

- Planuotojas peržvelgia READY procesų sąrašą, parenka procesą, kuris atrodo tinkamiausias vykdymui, ir perduoda jam procesorių.
- Tinkamiausias procesas vykdymui renkamas remiantis proceso prioritetu.
- **Proceso prioritetas** – proceso svarba, skalėje nuo 0 iki 255. 0 – žemiausia proceso svarba, 255 – aukščiausia proceso svarba.
- Procesorius bus suteiktas aukščiausią prioritetą turinčiam procesui.
- **PPS** - pasiruošusių procesų sąrašas

Planuotojo darbo algoritmas

Planuotojas naudoja prioritetinės eilės duomenų struktūrą, kad nuspręstų, kuriam sekančiam procesui paskirti procesoriaus darbą.

1. Iškviečiamas planuotojas.
2. Planuotojas prideda ką tik vykdytą procesą į prioritetinę eilę su pradiniu (baziniu) prioritetu.
3. Planuotojas peržiūri procesus. Visus READY procesus, kurie nėra prioritetinėje eilėje, prideda į prioritetinę eilę, o visus prioritetinėje eilėje esančius procesus, kurie nėra READY, išima iš prioritetinės eilės.
4. Planuotojas paima pirmą prioritetinėje eilėje esantį procesą ir pašalina jį iš eilės. Procesas *IdleProc* užtikrina, kad prioritetinėje eilėje visada bus nors vienas READY procesas.
5. Planuotojas padidina visų likusių procesų prioritetą vienetu (ne daugiau nei iki 255).
6. Planuotojas pratęsia pasirinkto proceso darbą (suteikia jam procesoriaus resursą).

Proceso deskriptorius

- OS kiekvienam procesui priskiria proceso deskriptorių
- Proceso deskriptorius saugo informaciją apie procesą:
  - **NAME** – proceso išorinis vardas
  - **CPU** – procesoriaus būsena
  - **RS -** turimų resursų sąrašas
  - **ST** – proceso būsena
  - **T –** proceso tėvas
  - **VS -** vaikinių procesų sąrašas
  - **BPR** – bazinis (pradinis) proceso prioritetas
  - **PR** – dabartinis proceso prioritetas

Proceso primityvai

Veiksmai, skirti darbui su procesais:

- **Kurti procesą**

```
#########################################################################
# kurti_procesa(
#   name, // proceso išorinis vardas
#   rs    // resursų sąrašas
# ):
#   pid = naujas_proceso_id();
#   ppid = dabartinis_proceso_id();
#   NAME[pid] = name;
#   CPU[pid] = tuscia_cpu_busena();
#   RS[pid] = rs;
```

```
#   ST[pid] = READYS;
#   PPS.append(pid);
#   T[pid] = ppid;
#   VS[pid] = [];
#   VS[ppid].append(pid);
#   BPR[pid] = pradinis_prioritetas();
#########################################################################
```

- **Naikinti procesą**

```
#########################################################################
# naikinti_procesa(
#   name  // proceso išorinis vardas
# ):
#   pid = surasti_pid_pagal_varda(name);
#   if ST[pid] == RUNNING then ATLAISVINTI_PROCESORIU(pid);
#   for cpid in VS[pid] do naikinti(cpid);
#   for r in RS[pid] do atlaisvinti(r);
#   atlaisvinti_proceso_deskriptoriu(pid);
#########################################################################
```

- **Aktyvuoti procesą**

```
#########################################################################
# aktyvuoti_procesa(
#   name // proceso išorinis vardas
# ):
#   pid = surasti_pid_pagal_varda(name);
#   if ST[pid] == READYS
#     then ST[pid] = READY

#   else if ST[pid] == BLOCKEDS
#     then ST[pid] = BLOCKED

#
#   if ST[pid] == READY then PLANUOTOJAS();
#########################################################################
```

- **Keisti proceso prioritetą**

```
#########################################################################
# keisti_proceso_prioriteta(
#   name,       // proceso išorinis vardas
#   naujas_pr  // naujas proceso prioritetas
# ):
#   pid = surasti_pid_pagal_varda(name);
#   senas_pr = PR[pid];
#   BPR[pid] = naujas_pr;
#   PR[pid] = naujas_pr;
#   if ST[pid] == READY and senas_pr < naujas_pr then PLANUOTOJAS();
#########################################################################
```

- **Pristabdyti procesą**

```
#########################################################################
# pristabdyti_procesa(
#   name // proceso išorinis vardas
# ):
#   pid = surasti_pid_pagal_varda(name);
```

```
#   st = ST[pid];
#   if st == RUNNING then ATLAISVINTI_PROCESORIU(pid);
#
#   if st == BLOCKED then ST[pid] = BLOCKEDS;
#   else if st == READY then ST[pid] = READYS;
#
#   if st == RUNNING then PLANUOTOJAS();
#########################################################################
```

Resursai

- Resursas – tai, dėl ko varžosi procesai.
- Resursai skirstomi į:
    - Statinius: šie resursai kuriami sistemos inicializavimo metu. Jie nėra naikinami sistemos darbo metu. Modeliuojamos sistemos atveju jie yra:
        - Procesorius
        - Atmintis
        - Fokusas
        - Klaviatūros įvestis
        - Terminalas
    - Dinaminius: šie resursai kuriami ir naikinami sistemos darbo metu. Šie resursai yra naudojami kaip pranešimai.

Resurso aprašas

- **NAME** – resurso išorinis vardas
- **LPS** - laukiančių procesų sąrašas
- **K** – resurso kūrėjas
- **PASK** – resurso paskirstytojas
- **LD** - laisvų resurso dalių sąrašas
- **PNR** – ar pakartotinio naudojimo resursas

Resursų lentelė

Pavadinimas	**Tipas**	**Kūrėjas**	**Dalys**
Procesorius	Statinis	StartStopProc	-
Atmintis	Statinis	StartStopProc	Laisvi atminties blokai
Fokusas	Statinis	StartStopProc	Sufokusuoto proceso ID
Klaviatūros įvestis	Statinis	StartStopProc	Įvestas simbolis
Terminalas	Statinis	StartStopProc	-
OS išjungta	Dinaminis	CLI	-
Pertraukimas	Dinaminis	VM	-
Iš Interrupt	Dinaminis	InterruptProc	Pertraukimo kodas
Užduotis atmintyje	Dinaminis	CLI, JobGovernor	END
Failo prieiga	Dinaminis	Paskirstytojas	Failo pavadinimas
Failo operacija	Dinaminis	CLI, JobGovernor	Nuoroda į failo prieiga, failo operacijos tipas, atminties blokas, kurį įrašyti, atminties bloko dydis

Failas įrašytas	Dinaminis	FileWriterProc	EXIT_CODE, įrašytų baitų skaičius
Failas nuskaitytas	Dinaminis	FileReaderProc	EXIT_CODE, nuskaitytų baitų skaičius
Failas sukurtas	Dinaminis	FileCreatorProc	EXIT_CODE
Failas ištrintas	Dinaminis	FileDeletorProc	EXIT_CODE

Resurso paskirstytojas

- Prašydamas resurso ar norėdamas jį atlaisvinti, procesas kreipiasi į atitinkamą resursų paskirstytoją
- Paskirstytojo paskirtis – suteikti paprašytą resursą procesui

Atminties resurso paskirstytojas

Atsakingas už atminties valdymą. Procesai ir kiti paskirstytojai turi kreiptis į šį paskirstytoją, kad gautų paskirtą atminties bloką, kurį galėtų naudoti savo paskirtims.

Atmintis skirstoma blokais po 4 kB. Vienas kreipimasis suteikia vieną bloką.

Jei nebėra laisvos atminties, atminties paskirstytojas blokuoja atminties prašantį procesą tol, kol kažkiek atminties atsilaisvins. Jei yra keli procesai, laukiantys atminties resurso, atminties resursas suteikiamas procesui, su aukščiausiu prioritetu. Jeigu nėra laisvų atminties blokų, procesas yra blokuojamas, kol atminties blokas bus atlaisvintas.

Fokuso resurso paskirstytojas

Atsakingas už proceso sufokusavimą. Vienu metu yra sufokusuotas tik vienas procesas. Sistemos inicializavimo metu *StartStopProc* sufokusuoja *CLI* procesą. Tuomet naudotojas gali sufokusuoti savo sukurtą - vartotojišką - procesą arba iš vartotojiško proceso grąžinti fokusą *CLI* procesui.

Procesoriaus resurso paskirstytojas

Už procesoriaus resurso paskirstymą atsakingas planuotojas (aprašytas aukščiau).

Failo prieigos resurso paskirstytojas

Vienu metu tik vienas procesas gali dirbti su tuo pačiu failu. Norėdamas gauti prieigą prie failo procesorius turi kreiptis į failo prieigos resurso paskirstytoją su norimo failo pavadinimu. Jei tuo metu failas nėra naudojamas kito proceso, procesui suteikiama nuoroda į failą, leidžianti manipuliuoti failo turiniu ir jį skaityti. Jei failas yra užimtas kito proceso, failo prašantis procesas yra blokuojamas tol, kol tas kitas procesas atlaisvins failo resursą ar pabaigs savo darbą.

Taip pat failo prieigos resurso paskirstytojas turi sąrašą sisteminių OS failų, prie kurių prieiga nėra suteikiama.

Jei prašoma sukurti failą ar ištrinti seną failą ir jo vietoje sukurti naują, šis paskirstytojas kreipiasi į FileManagerProc tam, kad šis procesas sukurtų ir ištrintų reikiamą failą ir tik tada gražina failo prieigos resursą.

Jei failas, prie kurio prašoma prieigos, yra neegzistuojantis ar sisteminis OS procesas, tada R2 registre reikšmė bus lygi nuliui, o R5 registras bus nustatytas į 0xFFFFFF, jei failas neegzistuoja (ir neprašoma sukurti naujo failo) arba 0xFEFEFE, jei failas yra sisteminis failas bei 0xFDFDFD, jei prašoma tik sukurti naują failą (neatidaryti seno), bet toks failas jau egzistuoja.

Klaviatūros įvesties resurso paskirstytojas

Procesas, norėdamas gauti klaviatūros įvestį turi kreiptis į klaviatūros įvesties resurso paskirstytoją. Tada jis yra blokuojamas, kol gauna klaviatūros įvestį. Procesas gali gauti klaviatūros įvesties resursą tik tada, kai

procesas yra sufokusuotas. Jeigu procesas, norintis gauti klaviatūros įvesties resursą, nėra sufokusuotas, tuomet jis yra blokuojamas tol, kol bus sufokusuotas.

Resurso primityvai

- **Prašyti resurso**
  - Blokuoja prašantį procesą, jei resursas nebuvo suteiktas
  - Gali suteikti resursą daugiau nei vienam procesui
  - Gali suteikti resursą nebūtinai prašančiam procesui dėl prioriteto

```
#########################################################################
# prasyti_resurso(
#   name, // resurso išorinis vardas
#   d     // resurso dalis
# ):
#   // įtraukiame procesą į laukiančiųjų sąrašą
#   rid = surasti_resursa_pagal_varda(name);
#   cpid = dabartinis_proceso_id();
#   LPS[rid].append(cpid, d);
#
#   // atblokuojame procesus, kuriems suteiktas resursas
#   L = PASKIRSTYTOJAS(rid);
#   for pid in L do:
#     atblokuoti_procesa(pid);
#
#   // blokuojame procesą, jeigu resursas nebuvo suteiktas
#   if not cpid in L then blokuoti_procesa(cpid);
#
#   PLANUOTOJAS();
#########################################################################
```

- **Atlaisvinti resursą**

```
#########################################################################
# atlaisvinti_resursa(
#   name, // resurso išorinis vardas
#   d     // resurso dalis
# ):
#   rid = surasti_resursa_pagal_varda(name);
#   LD[rid].add(d);
#
#   // atblokuojame procesus, kuriems suteiktas resursas
#   L = PASKIRSTYTOJAS(r);
#   for pid in L do:
#     atblokuoti_procesa(pid);
#
#   if L != [] then PLANUOTOJAS();
#########################################################################
```

- **Kurti resursą**

```
#########################################################################
# kurti_resursa(
#   name,   // resurso išorinis vardas
```

```
#   lps,    // laukiančių procesų sąrašas
#   pnr,    // ar pakartotinio naudojimo resursas
#   ld,     // resursą sudarančių dalių sąrašas
#   pask    // resurso paskirstytojas
# ):
#   pid = dabartinis_proceso_id();
#   rid = naujas_resurso_id();
#   NAME[rid] = name;
#   PNR[rid] = pnr;
#   K[rid] = pid;
#   LD[rid] = ld;
#   LPS[rid] = lps;
#   PASK[rid] = pask;
#   RS[cpid].add(rid);
#########################################################################
```

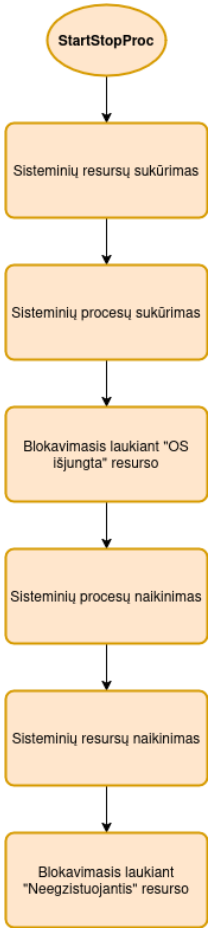
- **Naikinti resursą**

```
#########################################################################
# naikinti_resursa(
#   name // resurso išorinis vardas
# ):
#   rid = surasti_resursa_pagal_varda(name);
#   P = LPS[rid]; LPS[rid] = [];
#   for p in P do:
#     atblokuoti_procesa(p.pid);
#
#   naikinti_resurso_deskriptoriu(rid);
#   PLANUOTOJAS();
#########################################################################
```

MOS procesai

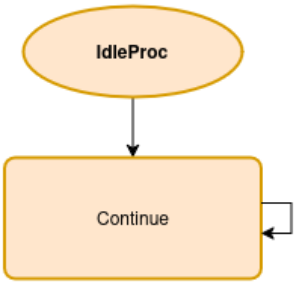
StartStopProc

- Šis procesas atsakingas už sistemos darbo pradžią ir pabaigą
- Šis procesas sukuriamas automatiškai įjungus kompiuterį
- Sukuria sisteminius resursus bei permanentinius procesus
- Gavęs signalą apie OS išjungimą, sunaikina procesus bei resursus



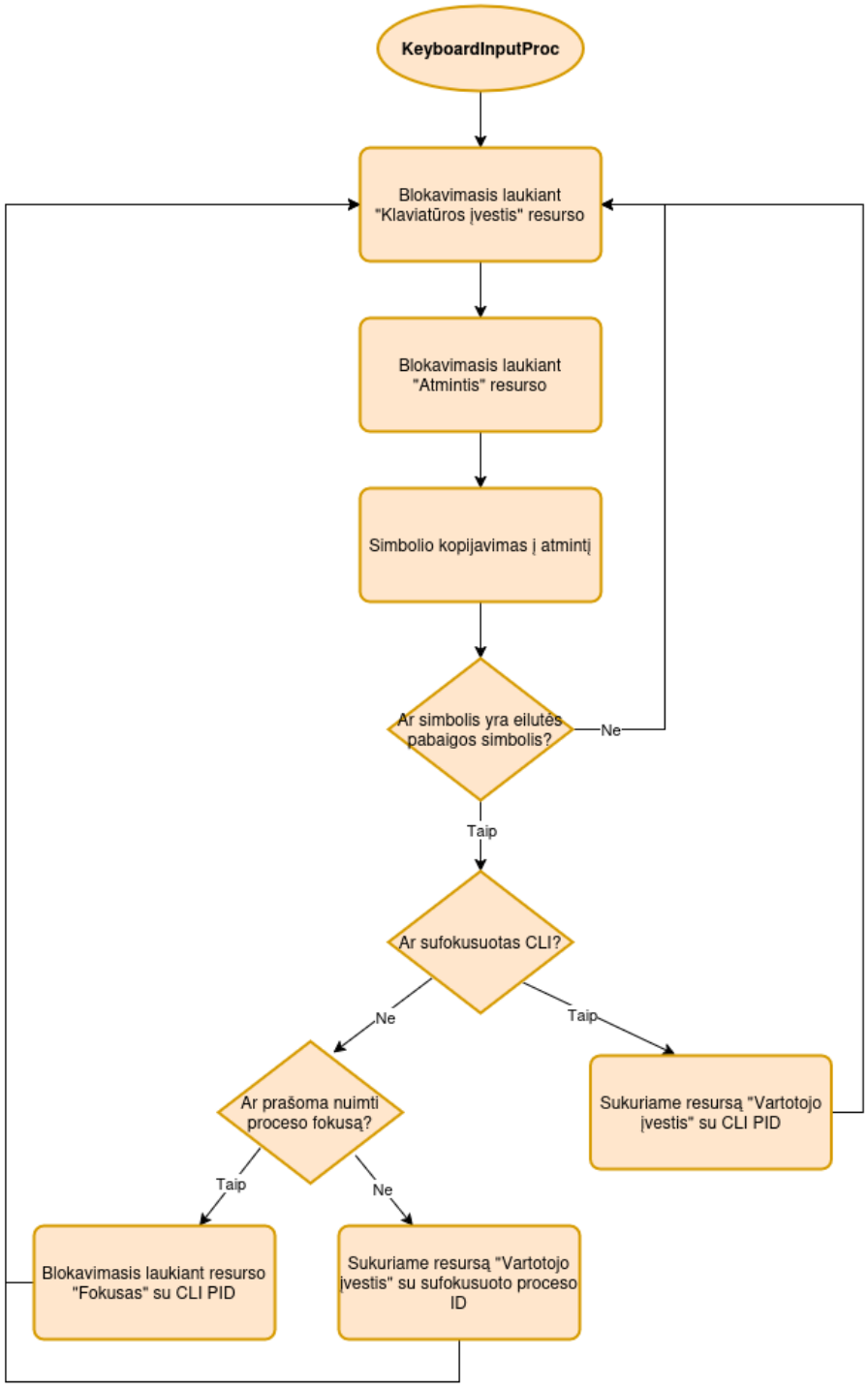
IdleProc

- Šis procesas užtikrina, kad sistemoje visada yra nors vienas procesas READY būsenoje
- Šis procesas sukuriamas *StartStopProc* proceso metu



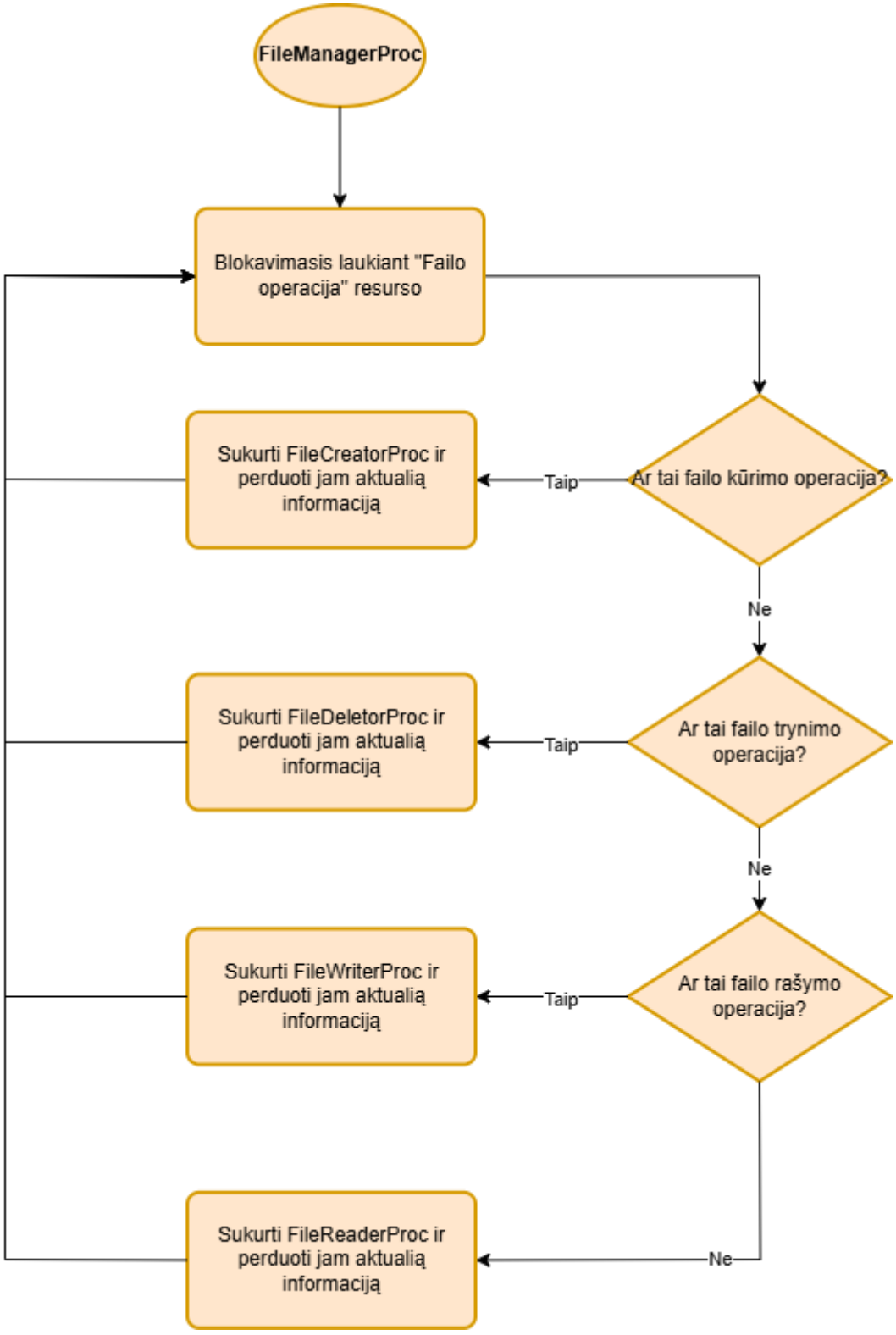
KeyboardInputProc

- Šis procesas atsakingas už klaviatūros įvesties apdorojimą
- Šis procesas sukuriamas *StartStopProc* proceso metu



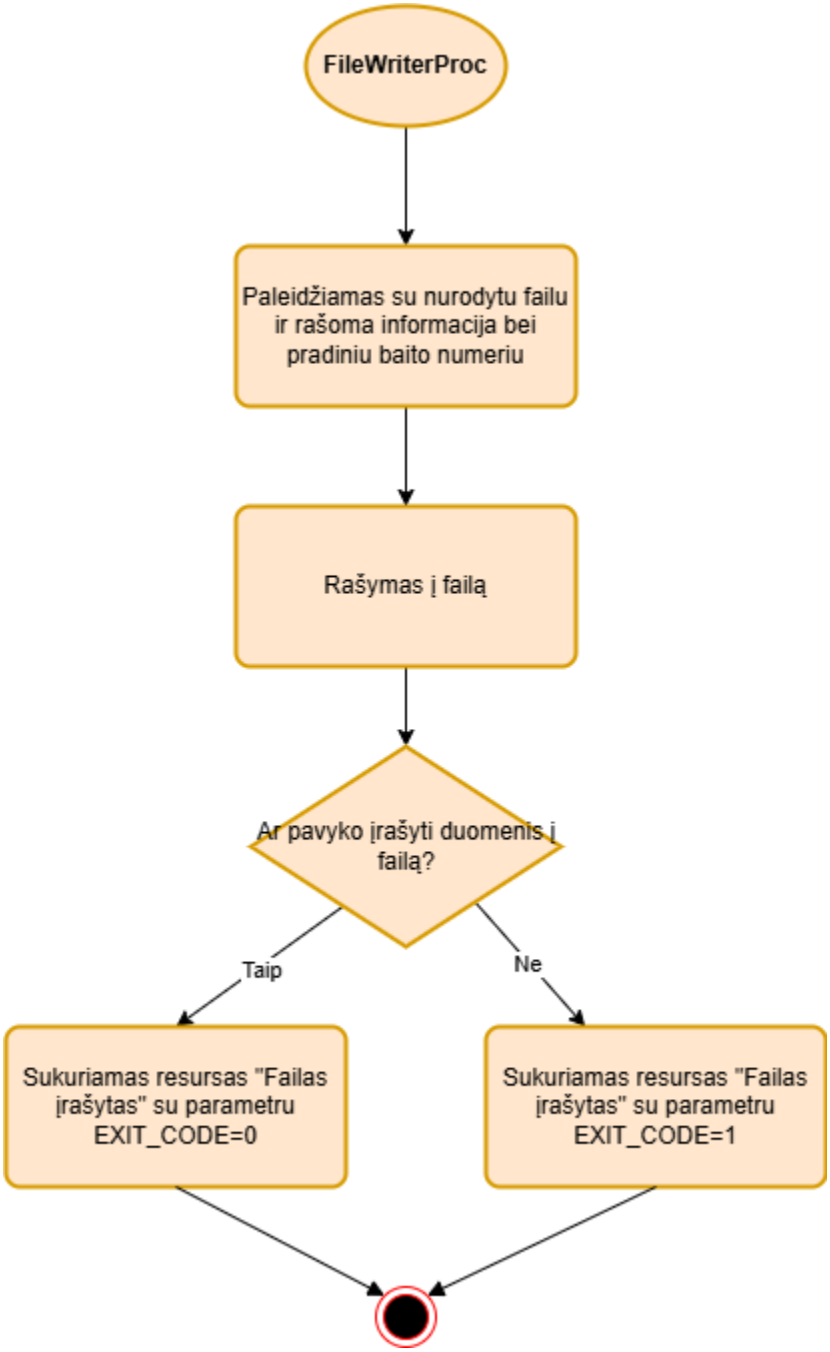
FileManagerProc

- Šis procesas atsakingas už darbą su failais.
- Šis procesas sukuriamas *StartStopProc* proceso.



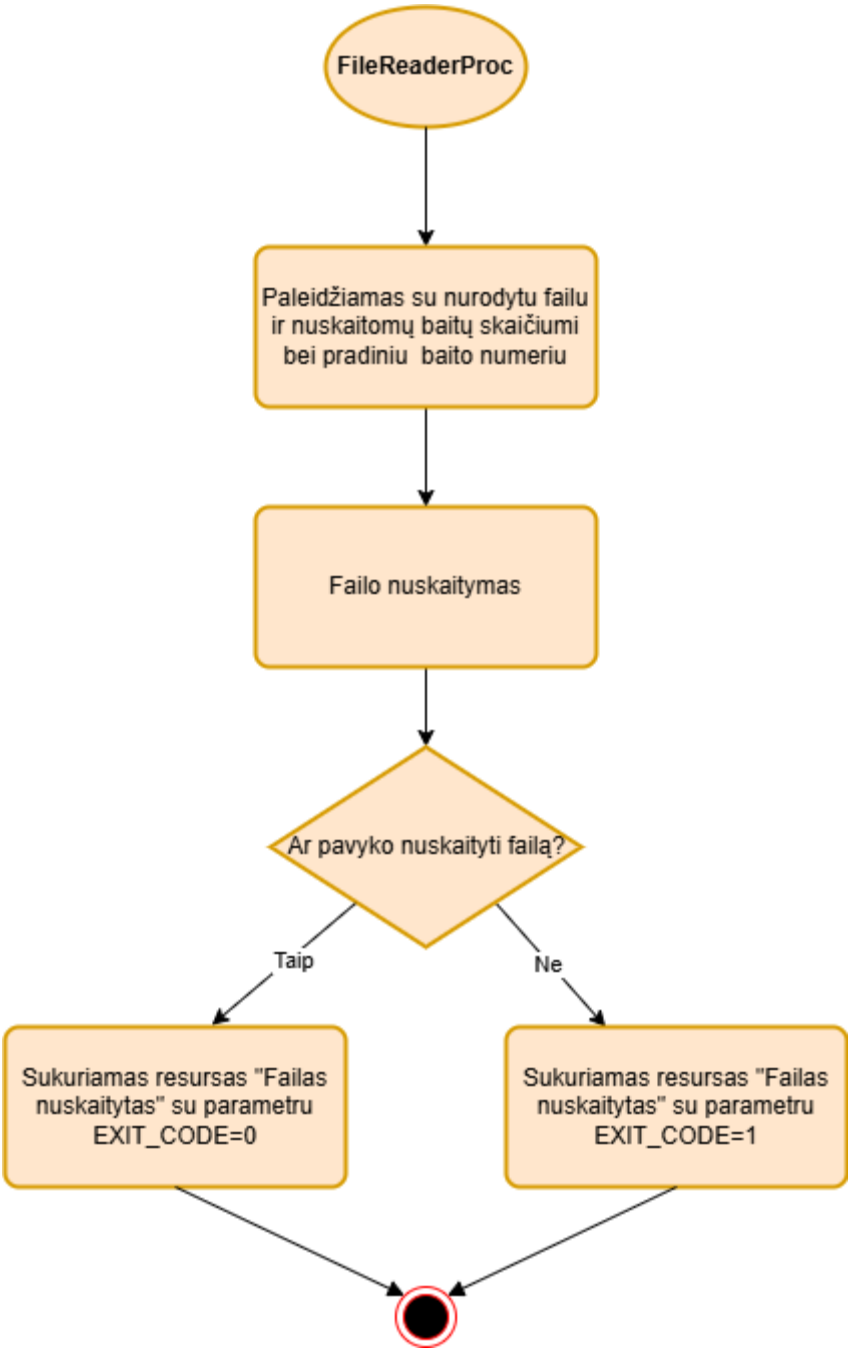
FileWriterProc

- Šis procesas atsakingas už duomenų rašymą į failą.
- Šis procesas sukuriamas *FileManagerProc* proceso.



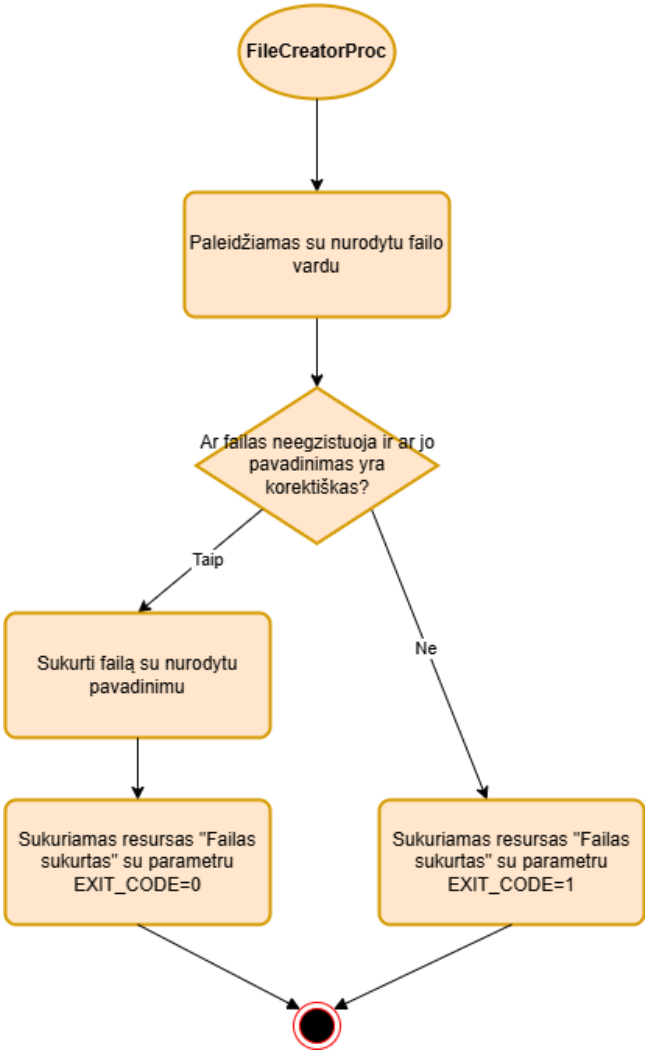
FileReaderProc

- Šis procesas atsakingas už duomenų skaitymą iš failo.
- Šis procesas sukuriamas *FileManagerProc* proceso.



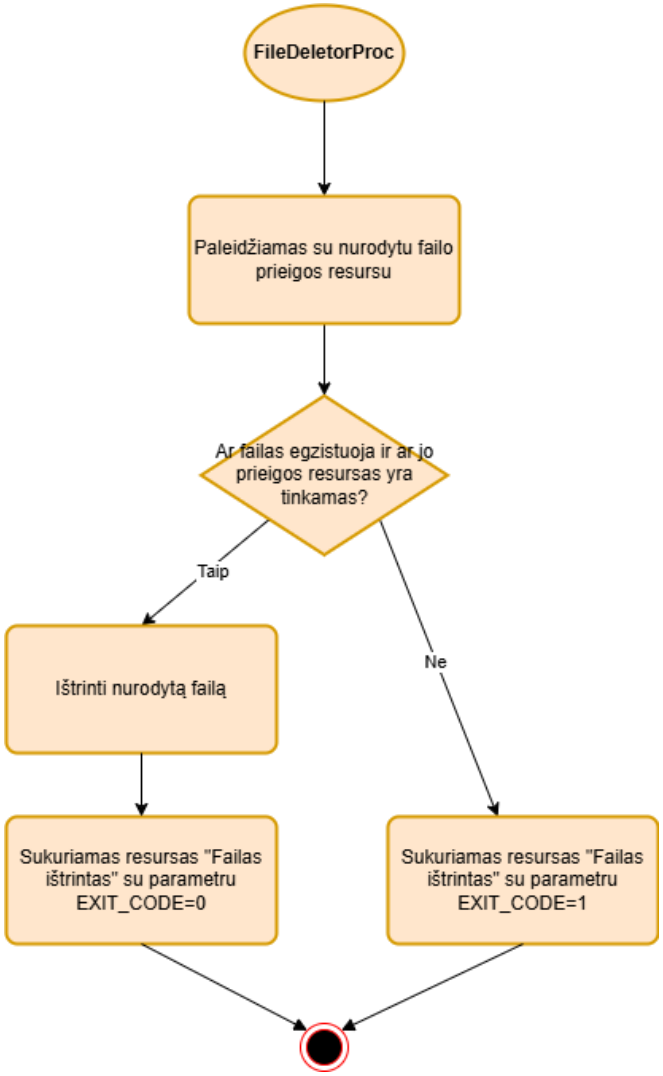
FileCreatorProc

- Šis procesas atsakingas už failo sukūrimą.
- Šis procesas sukuriamas *FileManagerProc* proceso.



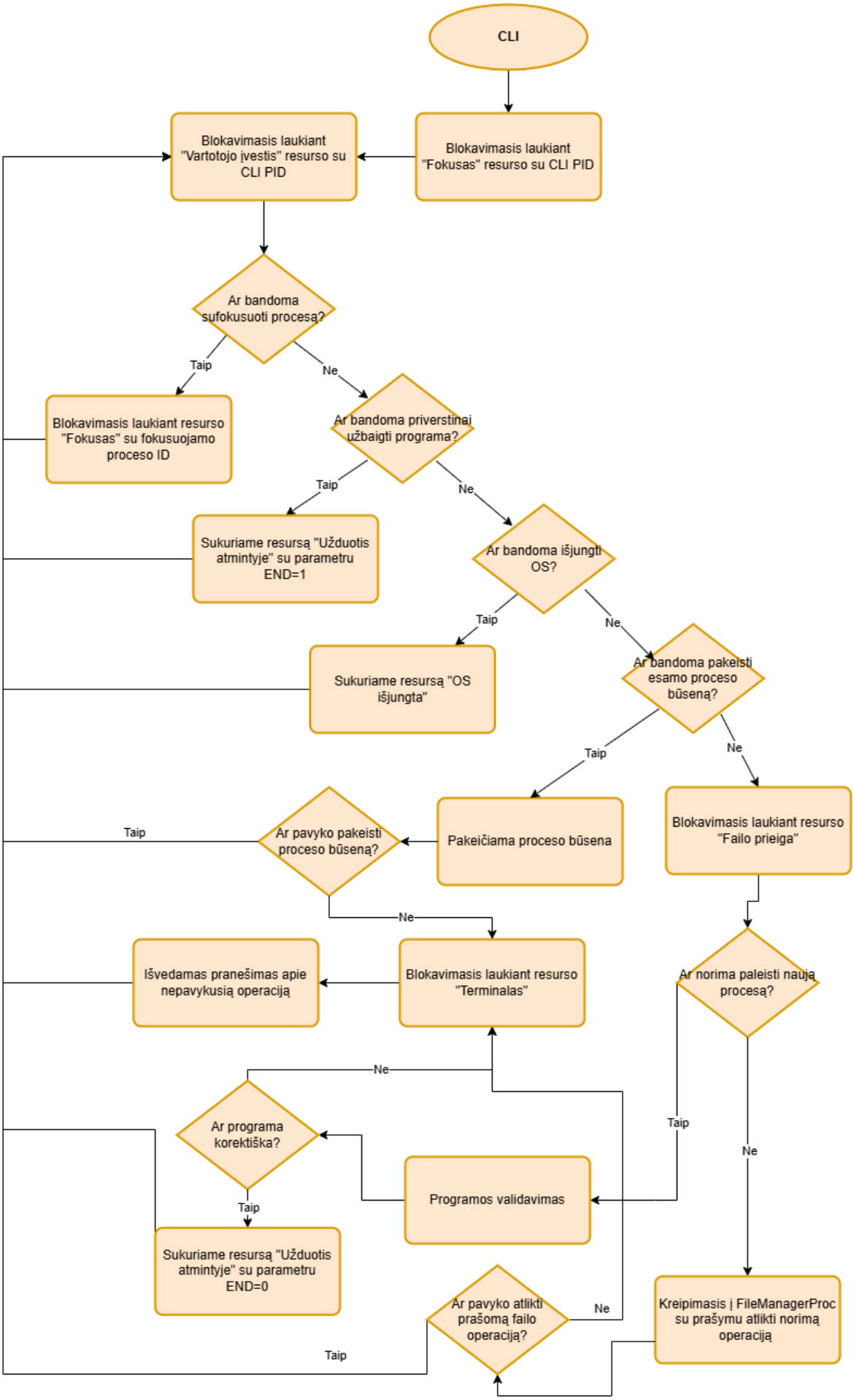
FileDeletorProc

- Šis procesas atsakingas už failo ištrynimą.
- Šis procesas sukuriamas *FileManagerProc* proceso.



CLI

- Šis procesas atsakingas už OS valdymą per naudotojo sąsają:
  - Programų sufokusavimą
  - Priverstinį programų užbaigimą
  - OS išjungimą
  - Naujų programų paleidimą
- Šis procesas sukuriamas *StartStopProc* proceso

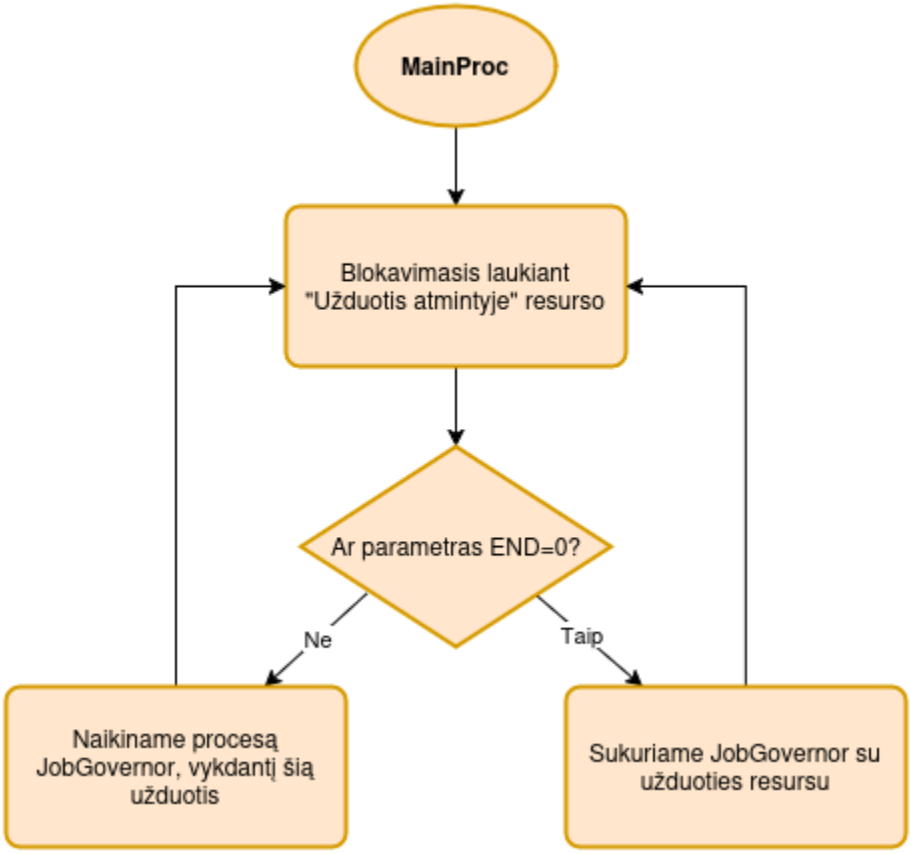

CLI
Blokavimasis laukiant "Fokusas" resurso su CLI PID
Blokavimasis laukiant "Vartotojo įvestis" resurso su CLI PID
Ar bandoma sufokusuoti procesą?
Taip
Ne
Blokavimasis laukiant resurso "Fokusas" su fokusuojamo proceso ID
Ar bandoma priverstinai užbaigti programa?
Taip
Ne
Sukuriame resursą "Užduotis atmintyje" su parametru END=1
Ar bandoma išjungti OS?
Taip
Ne
Sukuriame resursą "OS išjungta"
Ar bandoma pakeisti esamo proceso būseną?
Taip
Ne
Pakeičiama proceso būsena
Ar pavyko pakeisti proceso būseną?
Taip
Ne
Blokavimasis laukiant resurso "Failo prieiga"
Blokavimasis laukiant resurso "Terminalas"
Išvedamas pranešimas apie nepavykusią operaciją
Ar norima paleisti naują procesą?
Taip
Ne
Ar programa korektiška?
Ne
Taip
Programos validavimas
Sukuriame resursą "Užduotis atmintyje" su parametru END=0
Ar pavyko atlikti prašomą failo operaciją?
Ne
Taip
Kreipimasis į FileManagerProc su prašymu atlikti norimą operaciją

CLI galimybės

- Komanda *start <failo vardas>:*
  - Pabando paleisti procesą, kurio programos kodas yra pateiktame faile.
  - Jei įvyksta kokia klaida, apie tai praneša naudotojui.
- Komanda *process*
  - Išspausdina visų egzistuojančių procesų informaciją.
- Komanda *focus <proceso id>*
  - Liepia fokuso resurso valdytojui perduoti fokuso resursą iš CLI proceso į nurodytą procesą.
  - Jei nurodomas neteisingas pid, į terminalą išvedamas klaidos pranešimas.
- Komanda *kill <proceso id>*
  - Nužudo procesą
  - Negali nužudyti sisteminių procesų, tik naudotojo.
  - Jei nurodytas procesas yra sisteminis ar neegzistuojantis, į terminalą išvedamas klaidos pranešimas.
- Komanda *suspend <proceso id>*
  - Pristabdo procesą, jeigu procesas yra būsenoje READY arba BLOCKED
  - Jei nurodytas procesas yra sisteminis ar neegzistuojantis, į terminalą išvedamas klaidos pranešimas
- Komanda *unsuspend <proceso id>*
  - Aktyvuoja pristabdytą procesą, jeigu procesas yra būsenoje READYS arba BLOCKEDS
  - Jei nurodytas procesas yra sisteminis ar neegzistuojantis, į terminalą išvedamas klaidos pranešimas
- Komanda *shutdown*
  - Inicijuoja OS išjungimą
- Komanda *dir*:
  - Leidžia pamatyti kokie failai egzistuoja operacinėje sistemoje.
- Komanda *create <failo vardas>*
  - Sukuria tuščią failą su nurodytu vardu.
  - Jei toks failas jau egzistuoja, išspausdina klaidos pranešimą.
- Komanda *delete <failo vardas>*
  - Ištrina failą, su nurodytu vardu.
  - Negali ištrinti sisteminių (OS) failų, tik naudotojo sukurtus.
  - Jei toks failas neegzistuoja ar yra sisteminis, išmeta klaidos pranešimą.
- Komanda *write <failo vardas> <norimas įrašyti tekstas>*
  - Prirašo prie failo pabaigos bet kokį tekstą.
  - Norint parašyti specialius simbolius, galima naudoti \<*bet kokio simbolio ascii kodas dešimtainiais skaičiais*> šabloną.
  - Norint parašyti „\“ simbolį, vietoje to reikia rašyti „\\“.
- Komanda *display <failo vardas>*
  - Išspausdina failo turinį terminale ASCII formatu.
  - Jei toks failas nepasiekiamas ar neegzistuoja, išmetamas pranešimas naudotojui.

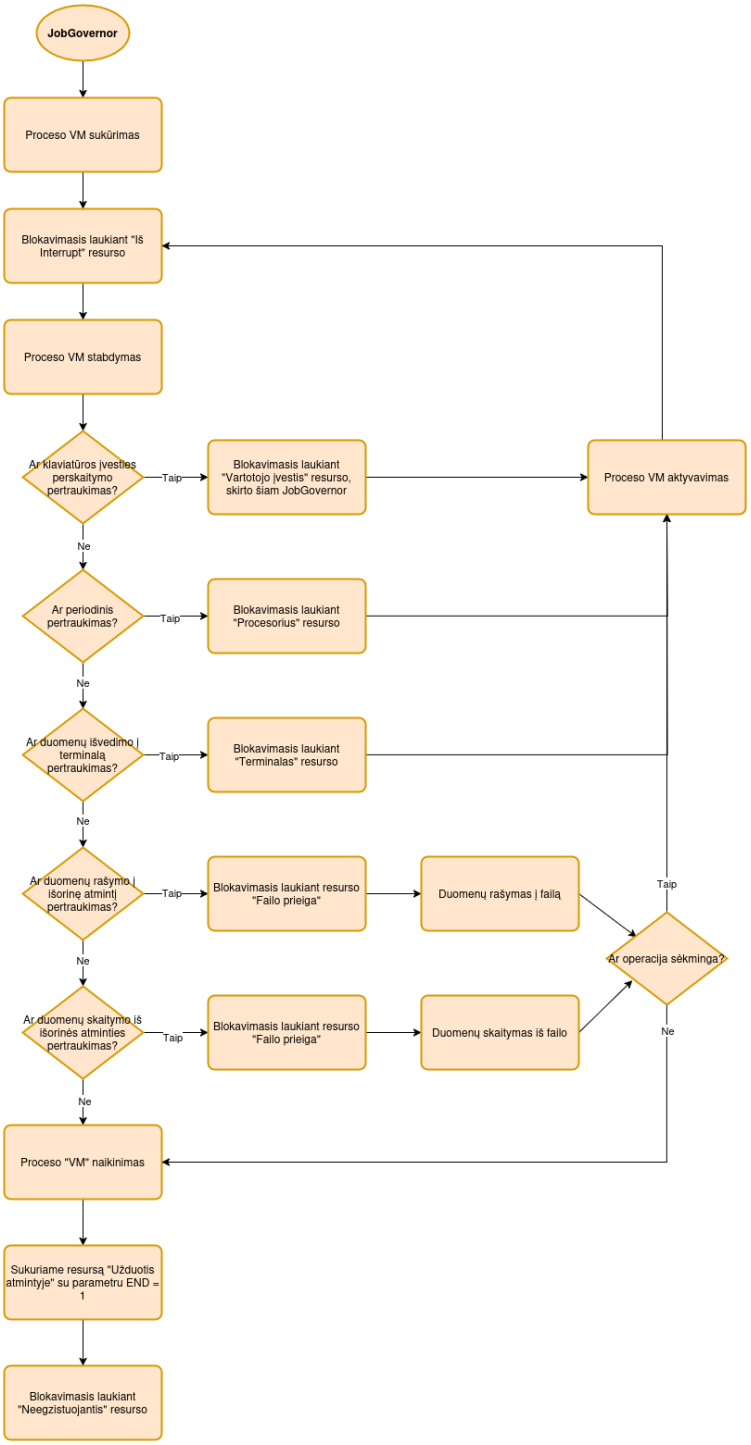
MainProc

- Šis procesas atsakingas už *JobGovernor* procesų kūrimą bei naikinimą: jeigu parametras END lygus 0, tuomet sukuriamas naujas *JobGovernor*, kitu atveju – sunaikinamas nurodytas *JobGovernor* procesas
- Šis procesas sukuriamas *StartStopProc* proceso



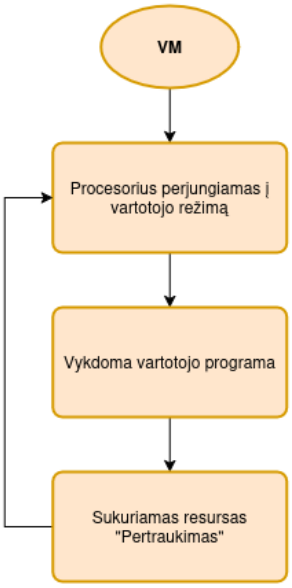
JobGovernor

- Šis procesas sukuria, naikina bei prižiūri *VM* procesą
- Prieš sukuriant VM procesą, alokuojama atmintis VM puslapių lentelei (4 MB), stekui (8 MB) ir programos kodui (dynamiškas dydis, nustatomas pagal programos dydį).
- Šis procesas sukuriamas *MainProc* proceso



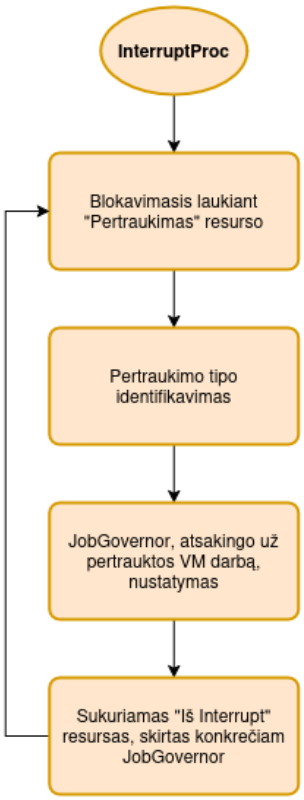
VM

- Šis procesas atsakingas už vartotojo programos vykdymą
- Šis procesas sukuriamas *JobGovernor* proceso



InterruptProc

- Šis procesas atsakingas už reagavimą į pertraukimus, kilusius virtualios mašinos darbo metu
- Šis procesas sukuriamas *StartStopProc* proceso



Klaviatūros įvestis

Procesas, norėdamas gauti klaviatūros įvesti turi pranešti operacinės sistemos resursų paskirstytojui, kad siekia gauti šią informaciją. Operacinė sistema, užfiksavusi klaviatūros įvestį, praneša apie tai resursų paskirstytojui, kuris tada pereina per visų procesų sąrašą, kurie yra pranešę, kad laukia klaviatūros įvesties, ir praneša jiems, kad buvo klaviatūros įvestis ir perduoda procesams jos reikšmę.

Klaviatūros įvesties atžvilgiu visi procesai skirstomi į du tipus: sufokusuotus ir nesufokusuotus. Tik sufokusuoti procesai gali gauti klaviatūros įvestį. Nesufokusuoti procesai yra blokuojami, kol gauna fokuso resursą.

Terminalo išvestis

Procesas, norėdamas išvesti ką nors į terminalą, turi kreiptis į terminalo resurso valdytoją su norimu išvesti tekstu.

Tada terminalo resursų valdytojas į šį kreipinį gali reaguoti dviem būdais:

1. Jei procesas, kuris siekia išvesti kažką į terminalą, turi fokuso resursą, tai šis tekstas iš karto yra išvedamas į terminalą.
2. Jei procesas neturi fokuso resurso, terminalo resurso valdytojas išsisaugo atsiųstą tekstą į konkrečiam procesui priklausantį laikiną atminties buferį. Šis tekstas bus išvestas į terminalą tuo momentu, kai procesas gaus fokuso resursą. Išvedus tekstą resursas pašalinamas. Jei procesas bando kažką išvesti į terminalą neturėdamas fokuso resurso ir jei nuo praeito karto, kai procesas bandė ką nors išvesti į terminalą, procesas taip ir nebuvo gavęs fokuso, tai naujas buferis nekuriamas, tiesiog papildomas senasis buferis.

OS pertraukimai

- INT 0:

  Sukeliamas įvykus dalybai iš nulio. Kilus šiam pertraukimui, OS nutraukia procesą, kuris sukėlė šį pertraukimą.

- INT 1:
  Sukeliamas naudotojui paspaudus klaviatūros klavišą. OS perduoda KeyboardInputProc procesui resursą „Klaviatūros įvestis“.
- INT 2:
  Sukeliamas, kai bandoma vykdyti operaciją, kurios op kodas yra nežinomas ar kuri yra negalima dėl semantinių sumetimų. OS nutraukia procesą, sukėlusį šį pertraukimą.
- INT 3:
  Sukeliamas, kai procesas, veikiantis virtualios mašinos režimu, bando pasiekti atminties adresą, kuris yra neprieinamas tam procesui. OS nutraukia procesą, sukėlusį šį pertraukimą.
- INT 4:
  Kreipimasis į terminalą. R3 registre saugomas adresas simbolių eilutės pradžios, kurią norima išvesti į terminalą.
- INT 5:
  Periodiškai kylantis pertraukimas. Iškviečiamas procesų planuotojas, kuris nusprendžia, kuriam procesui bus suteiktas procesoriaus resursas.
- INT 6:
  Kreipimasis į FileManagerProc, su prašymų įrašyti kažką į failą. R2 registre saugoma nuoroda į failo prieigos resursą. R3 registre saugoma nuoroda į atminties bloką, kurį norima įrašyti. R4 registre saugomas norimas įrašyti baitų skaičius. R5 registre pateikiamas pirmasis baitas, nuo kurio reikia pradėti rašyti. Jei R5 reikšmė didesnė nei bendras failo baitų skaičius, rašoma į failo pabaigą. R4 registre grąžinama reikšmė yra lygi įrašytų baitų skaičiui. Iškvietus šį pertraukimą OS kreipiasi į failo prieigos resurso paskirstytoją, kad

patikrintų, ar failo prieigos resursas yra egzistuojantis. Jei ne, grįžtama nustatant R4 registrą į 0 ir R2 registrą į 1.

- INT 7:
  Kreipimasis į FileManagerProc, su prašymu perskaityti kažką iš failo. R2 registre saugoma nuoroda į failo prieigos resursą. R3 registre saugoma nuoroda į atminties bloką, į kurį norima įrašyti. R4 registre saugomas norimas nuskaityti maksimalus baitų skaičius, o R5 registre nurodoma vieta, nuo kurios norima skaityti. Jei faile yra mažiau, nei nurodytas maksimalus baitų skaičius + pradinio baito numeris, nuskaitomi visi failo baitai ir R4 registro reikšmė pakeičiama į nuskaitytų baitų skaičių. Iškvietus šį pertraukimą OS kreipiasi į failo prieigos resurso paskirstytoją, kad patikrintų, ar failo prieigos resursas yra egzistuojantis. Jei ne, grįžtama nustatant R4 registrą į 0 ir R2 registrą į 1.
- INT 8: Kreipimasis į failo prieigos resurso valdytoją prašant suteikti prieigos prie kokio nors failo resursą. R1 registre saugomas failo pavadinimas, prie kurio norima gauti prieigą. R5 registre saugoma reikšmė, kuri nurodo, kaip norima elgtis su failu, kuriam prašoma prieigos. Jei failo prieigos resurso valdytojas suteikia prieigą prie failo, R2 registre jis grąžina nuorodą į failo prieigos resursą, suteikiantį teisę rašyti į tą failą. Galimos R5 reikšmės:
  - 0 – prašoma atidaryti failą tik jei jis egzistuoja. Rašomi duomenys bus rašomi vietoje senų duomenų.
  - 1 – prašoma atidaryti failą tik jei jis egzistuoja. Rašomi duomenys bus priduriami failo pabaigoje.
  - 2 – prašoma sukurti failą tik jei jis neegzistuoja.
  - 3 – prašoma atidaryti egzistuojantį failą arba sukurti failą jei jis neegzistuoja. Jei failas egzistuoja, duomenys bus rašomi vietoje senų duomenų.
  - 3 – prašoma atidaryti egzistuojantį failą arba sukurti failą jei jis neegzistuoja. Jei failas egzistuoja, duomenys bus priduriami failo pabaigoje.
- INT 9: Kreipimasis į failo prieigos resurso valdytoją pranešant, kad norima atlaisvinti tam tikro failo prieigos resursą. R2 registre turi būti nuoroda į failo prieigos resursą. Failo prieigos resurso valdytojas pažymi šį resursą kaip atlaisvintą.
- INT 10: Kreipimasis į FileManagerProc prašant ištrinti failą iš sistemos. R2 turi būti pateikiama nuoroda į failo prieigos resursą. FileManagerProc ištrina failą, susijusį su resursu ir pažymi šį resursą kaip atlaisvintą.
- INT 11: Kreipimasis į atminties resurso valdytoją prašant išskirti atminties bloką. Jeigu atminties resurso valdytojas suteikia atmintį, R3 registre įrašomas 0, o R2 registre įrašomas išskirto atminties bloko pradžios adresas. Kitu atveju, R3 registre įrašomas 1.
- INT 12: Kreipimasis į atminties resurso valdytoją prašant atlaisvinti išskirtą atminties bloką. R2 turi būti pateikiamas išskirtos atminties pradžios adresas. Jeigu atminties resurso valdytojas suteikia atminties bloką, R3 registre įrašomas 0. Kitu atveju, R3 registre įrašomas 1.

Jei pertraukimo metu sunaikinamas procesas, kuris turėjo fokuso resursą, fokuso resursas yra suteikiamas CLI procesui. Jei sunaikinamas procesas yra CLI procesas, tada CLI procesas yra paleidžiamas iš naujo.

Failų sistema

Operacinėje sistemoje visa naudotojo ilgalaikė informacija (ir dalis pačios sistemos informacijos) saugoma failų pavidalu. Failai yra saugomi išorinėje atmintyje, už jų išsidėstymą joje yra atsakinga operacinė sistema.

Kiekvienas failas turi savo unikalų vardą, pagal kurį į jį galima kreiptis ir galimą jį atpažinti. Failai šioje operacinėje sistemoje nėra skirstomi į jokius katalogus ar direktorijas. Visi jie yra saugomi bendrame lygmenyje.

Nėra jokių failų vardų standartų, svarbu, kad jie tiesiog nesikartotų. Failų vardai turi būti sudaryti tik iš abėcėlės didžiųjų ir mažųjų raidžių, skaičių bei simbolių ()_. -. Failų varduose negali būti tarpų ar kitų „whitespace“ simbolių. Failo vardas negali būti ilgesnis nei 255 simboliai ir ne trumpesnis nei 1 simbolis.

Operacinė sistema tik sau pasiekiamoje atmintyje saugo lentelę, su visų failų vardais bei tų failų vietomis atmintyje. Kiekvienas failas gali būti nebūtinai vienoje atminties vietoje, jis gali būti išskaidytas dalimis.

Taip pat operacinė sistema saugo sąrašą failų, kurie yra operacinės sistemos failai. Šis sąrašas yra pasiekiamas kitiems operacinės sistemos procesams, ir leidžia nustatyti, ar tam tikros operacijos su kai kuriais failais yra galimos.

OS išjungimas

Paskelbus, kad norima išjungti operacinę sistemą, atliekami šie veiksmai:

1. Nebeleidžiama sukurti jokių naujų procesų.
2. Sustabdomi ir sunaikinami naudotojo procesai.
3. Palaukiama, kol visi FileWriterProc, FileCreatorProc, bei FileDeletorProc pabaigia darbą. Jei jie užtrunka ilgiau nei 30 sekundžių, jie yra priverčiamai sustabdomi.
4. Sustabdomi ir sunaikinami sisteminiai procesai.
5. Atlaisvinami visi resursai.
6. Sunaikinami visi resursai.
7. Mašina pereina į nieko nedarymo būseną.