VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

3 KURSAS

Multi-OS projektas

Darbą atliko:

Justinas Mačiulis (KM grupė: 2)

Mantas Pipinė (KM grupė: 2)

Paulius Mitkus (KM grupė: 2)

Darius Turčinskas (MI grupė: 1)

Turinys

[1. Procesai 2](#_Toc448780080)

[1.1. Apie procesus 2](#_Toc448780081)

[1.2. Tipai 2](#_Toc448780082)

[1.3. Būsenos 2](#_Toc448780083)

[1.4. Planuotojas 3](#_Toc448780084)

[1.5. Prioritetai 4](#_Toc448780090)

[1.6. Primityvai 4](#_Toc448780091)

[2. Resursai 7](#_Toc448780092)

[2.1. Apie resursus 7](#_Toc448780093)

[2.2. Resursų lentelė 7](#_Toc448780097)

[2.3. Paskirstytojai 8](#_Toc448780102)

[2.4. Primityvai 8](#_Toc448780108)

[3. Deskriptoriai 10](#_Toc448780109)

[3.1. Procesų deskriptorius (PD) 10](#_Toc448780110)

[3.2. Procesoriaus deskriptorius (PRD) 11](#_Toc448780111)

[3.3. Resursų deskriptorius (RD) 11](#_Toc448780112)

[4. Procesų paketas 11](#_Toc448780116)

[4.1. Apžvalga 11](#_Toc448780121)

[4.2. StartStop 12](#_Toc448780127)

[4.3. ProgramToExtMem 13](#_Toc448780134)

[4.4. MainProc 13](#_Toc448780142)

[4.5. Reader 14](#_Toc448780151)

[4.6. Interrupt 14](#_Toc448780161)

[4.7. JobHelper 15](#_Toc448780172)

[4.8. Laukimas 15](#_Toc448780184)

[4.9. LoadProgram 16](#_Toc448780197)

[4.10. VirtualMachine 16](#_Toc448780211)

[4.11. Input 17](#_Toc448780239)

[4.12. Output 17](#_Toc448780267)

# Procesai

## Apie procesus

Procesas – tai programa esanti veiklumo stadijoj kartu su esamomis registrų reikšmėmis ir savo kintamaisiais. Tą veiklumo stadiją nusako procesoriaus aprašas – deskriptorius. Tame apraše ir yra laikomi visi reikalingi parametrai kaip virtualios mašinos registrų reikšmės ir kiti kintamieji.

Proceso aprašas yra dinaminis objektas, jis gali būti sukurtas ar sunaikintas vykdymo eigoje.

Pačius procesus taipogi kuria procesai, taigi, kad sukurti pirmuosius procesus reikia vieno pradinio proceso, kurį pavadinsime StartStop (apie jį šiek tiek vėliau).

Kiekvienas procesas turi virualų procesorių. Taip pat operacinėje sistemoje procesai „kovoja“ dėl resursų (apie kuriuos vėliau) bei žinoma procesų veikimas turi būti kiek įmanoma lygiagretesnis.

## Tipai

Procesai yra skirtomi į du tipus: **sisteminius** ir **vartotojo**. Sisteminiai aptarnauja vartotojus, o vartotojai tiesiog vykdo programą.

Sisteminiai procesai yra kuriami paleidžiant sistemą (būtent prieš tai minėtas StartStop ir sukurs juos), o naikinami taipogi atitinkamai su sistema.

Vartojo procesai yra sukuriami sisteminių procesų. Kartu, jei reikia, gali būti sukurti ir sisteminiai procesai reikalingi aptarnauti tą vartojo procesą.

Vartotojo procesai veikia lygiagrečiai dėl pertraukimų, o sisteminiai procesai – blokuodamiesi, t.y. laukdami kol galės atlikti savo darbą, kuris paprastai būna susijęs su vartotojo procesu.

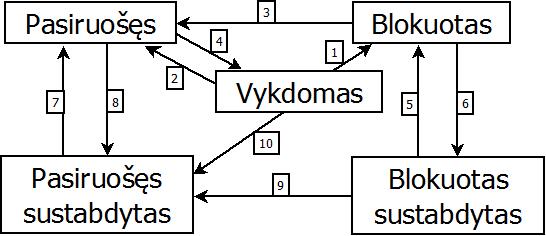
## Būsenos

Procesas gali gauti procesorių tik tada, kai jam netrūksta jokio kito resurso. Procesas gavęs procesorių tampa vykdomu. Procesas, esantis šioje būsenoje, turi procesorių, kol sistemoje neįvyksta pertraukimas arba einamasis procesas nepaprašo kokio nors resurso (pavyzdžiui, prašydamas įvedimo iš klaviatūros). Procesas blokuojasi priverstinai (nes jis vis tiek negali tęsti savo darbo be reikiamo resurso). Tačiau, jei procesas nereikalauja jokio resurso, iš jo gali būti atimamas procesorius, pavyzdžiui, vien tik dėl to, kad pernelyg ilgai dirbo.

Procesų būsenos gali būti tokios:

* Vykdomas – turi procesorių,
* Blokuotas – prašo resurso (išskyrus procesorių),
* Pasiruošęs – turi visus reikalingus resursus ir vienintelis trūkstamas resursas yra procesorius,
* Sustabdytas – kito proceso sustabdytas procesas.

Diagrama, vaizduojanti, kaip procesas gali pakliūti i tam tikrą būseną ir iš jos išeiti:



Yra 10 perėjimų:

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

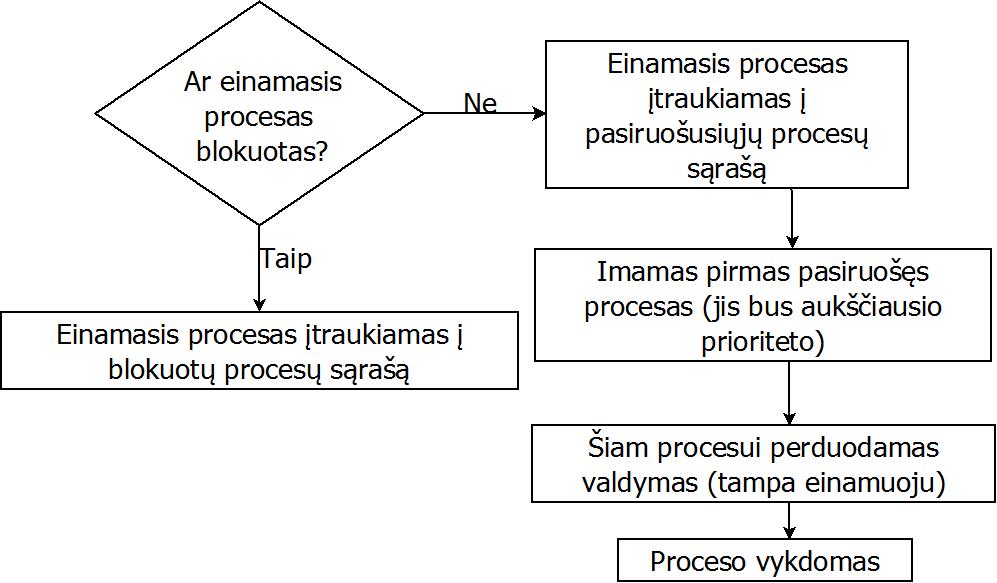
## Planuotojas

Planuotojas – centrinio procesoriaus resursų paskirstytojas.

Planuotojo užduotis – skirstyti procesorių. Atimti procesorių iš vienų procesų ir duoti jį kitiems, jo manymu, labiausiai vertiems procesoriaus. Planuotojo tikslai:

* užtikrinti, kad kiekvienas procesas gautų procesorių reikiamą laiko tarpą;
* maksimaliai užimti procesorių;
* iki minimumo sumažinti atsakymo laiką varotojams.

Planuotojas kviečiamas, kai norima procesorių perduoti kitam procesui. Jo veiksmų seką galima būtų pavaizduoti tokia diagrama:



Pirmas planuotojo žingsnis yra einamojo proceso būsenos tikrinimas. Jei einamasis procesas nėra blokuotas, jis įtraukiamas į pasiruošusių procesų sąrašą.

Sekantis planuotojo žingsnis yra tikrinimas, ar yra pasiruošusių procesų. Jeigu jų nėra, tai tada bus vykdomas specialiai įdėtas procesas su mažiausiu įmanomu prioritetu (Laukimas). Jis tiesiog užims procesoriaus laiką, kol atsiras kitas pasiruošęs procesas. Jeigu pasiruošusių procesų sąrašas nėra tuščias, tai imamas pirmasis sąrašo elementas su aukščiausiu prioritetu - proceso aprašas. Sekantis žingsnis – valdymo perdavimas. Proceso apraše laikoma virtualaus procesoriaus būsena priskiriama realiam procesoriui, išsaugoma einamojo proceso aplinka, užkraunama pasirinktojo proceso aplinka. Naujasis procesas pažymimas kaip einamasis.Viskas. Pakeistoje proceso aplinkoje bus ir grįžimo į naująjį einamąjį procesą adresas. Taigi, valdymas netiesiogiai perduotas. Baigdama savo darbą, planuotojo funkcija grįš ne ten, iš kur atėjo, o ten, kur rodo užkrauta einamojo proceso aplinka.



## Prioritetai

Proceso prioritetas – tai požymis, parodantis proceso svarbumą. Kuo mažesnė reikšmė, tuo procesas svarbesnis (maksimalus svarbumas = 0).

|  |  |
| --- | --- |
| **Procesų prioritetai** | |
| Proceso prioritetas | Proceso(-ų) pavadinimas(-ai) |
| 0 | StartStop |
| 1 | Interrupt; LoadProgram; ProgramToExtMem; Input; Output; MainProc, Reader |
| 2 | JobHelper |
| 3 | VirtualMachine |
| 4 | Laukimas |

## Primityvai

Procesų primityvų paskirtis – pateikti vienodą ir paprastą vartotojo sąsają darbui su procesais. Kiekvieno primityvo programos gale kviečiamas planuotojas. Yra 4 primityvai:

* **Kurti procesą**. Šis primityvas naudojamas, norint sukurti procesą, faktiniais parametrais nurodant tokias kuriamo proceso komponentes:
  + **n** – išorinis vardas;
  + **s0 –** kuriamo proceso procesoriaus pradinė būsena;
  + **M0 –** vartotojo atminties pradinė būsena (kiek išskirta vartotojo atminties resursų);
  + **R0 –** kiti išskiriami resursai;
  + **k0 –** proceso prioritetas;
  + **Algoritmas:**
    - BEGIN
    - i := NVV; (reikia nustatyti ir gauti proceso VV. NVV grąžina naują numerį)
    - ID [i] := n; (proceso ID, išorinis vardas, ryšiams tarp procesų nurodyti)
    - CPU[i] := s0; (užrašoma procesoriaus pradinė būsena deskriptoriuje)
    - OA[i] := M0; (turimas vartotojo atminties kiekis)
    - R[i] := R0; (kitų turimų proceso resursų komponentė (nuoroda į sąrašą))
    - PR[i] := k0; (priskiriamas proceso prioritetas)
    - ST[i] := READY; (laikysime, kad procesas sukuriamas su status “pasiruošęs”)
    - SD[i] := PPS; (priskiriam procesą PPS’ui)
    - T[i] := \*; (einamasis procesas bus žymimas „\*”. Tai einamojo proceso VV)
    - S[i] := X; (procesas turi nuorodą į sūnų sąrašą. Iš pradžių jis tuščias)
    - Įjungti(S[\*], i); (operuojame sąrašais su antraštėmis, o antraštės – tai programos, dirbančios su sąrašais. Dabar dirbantis (einamasis) procesas įgijo sūnų. Reikia papildyti jo sūnų sąrašą)
    - Įjungti(PPS, i); (naują procesą reikia įtraukti į pasiruošusių procesų sąrašą)
    - END
* **Naikinti procesą**. Procesas gali sunaikinti bet kurį savo palikuonį. Naikinant procesą reikia sunaikinti visus jo vaikus, kad nebūtų nevaldomų procesų (kad nebūtų chaoso). Parametrai:
  + **n** – naikinamo proceso išorinis vardas.
  + **Algoritmas**:
    - BEGIN
    - kviestiPlanuotoja := false; (pagalbinis loginis parametras, kuris nurodo ar reikia iškviesti planuotoją)
    - i := VV(n) (pagal išorinį vardą (proceso ID) nustatomas proceso VV)
    - p := T[i]; (nuoroda į tėvo procesą pasiemama)
    - Pašalinti (s[p], i); (pašalinamas procesas iš tėvo vaikų sąrašo)
    - NUTRAUKTI(i)
    - if ST[i] = RUN THEN  
       STOP(i);  
       kviestiPlanuotoja := true;
    - Pašalinti (SD[i], i); (kadangi procesas visada yra kažkuriame sąraše, tai reikia jį iš ten pašalinti. SD saugo nuorodą į sąrašą, iš kurio reikia pašalinti, i - VV)
    - FOR ALL s iš s[i] DO NUTRAUKTI(s); (nutraukiame i-tojo proceso sūnų vykdymą)
    - FOR ALL r iš OA[i] ir R[i] DO   
       IF PNR THEN ĮJUNGTI(PA[r], Pašalinti(OA[i] ir R[i]))  
      (r – resursai. Atlaisviname pakartotinio naudojimo resursus)
    - FOR ALL r iš SR[i] DO NRD(r); (liko sunaikinti proceso sukurtų resursų deskriptorius)
    - NPD(i); (naikiname proceso deskriptorių)
    - END
* **Stabdyti procesą**. Keičiama proceso būsena iš blokuotos į blokuotą sustabdytą arba iš pasiruošusios į pasiruošusią sustabdytą. Einamasis procesas stabdomas tampa pasiruošusiu sustabdytu. Šiuo atveju stabdomas vienas procesas. Kiti – sūniniai – tęsia darbą. Parametrais nurodome:
  + **n** – išorinis vardas.
  + **a** – atsakymų srities adresas į kurį grąžinama informacija apie stabdomo deskriptoriaus būseną (jo kopija)
  + **Algoritmas**:
    - BEGIN
    - i := VV(n); (pagal išorinį vardą (proceso ID) nustatomas proceso VV)
    - IF (ST[i] = RUN) THEN STOP(i); (Stabdomas procesas gali būti “Vykdomas”, “Pasiruošęs”, “Blokuotas”. Pirmiausia iš jo reikia atimti procesorių. STOP(i) pertraukia procesorių, kuris vykdo i-tąjį procesą. Tai procesorius P[i] turi prisiminti pertraukto procesoriaus būseną į CPU[i] ir pasakyti, kad procesorius vykdęs i-tąjį procesą yra atlaisvintas: PROC[P[i]] = X)
    - a := copydesk[i]; (ši procedūra padaro deskriptoriaus kopiją)
    - IF (ST[i] = BLOCK) THEN   
       ST[i] := BLOCKS   
      ELSE IF (ST[i] = READY) THEN   
       ST[i] := READYS;   
      (koreguojamas i-tojo proceso būsena)
    - IF ST[i] = RUN THEN   
       ST[i] := READYS  
       PLANUOTOJAS;   
      (jei buvo pristabdytas vykdomas procesas, tai reikia iškviesti planuotoją, nes galbūt kiti procesai laukia procesoriaus. Planuotojo vykdymui naujas procesas nesukuriamas)
    - END
* **Aktyvuoti procesą**. Keičiama proceso būsena iš blokuotos sustabdytos į blokuotą, ar pasiruošusios sustabdytos į pasiruošusią. Parametras:
  + n – išorinis vardas (proceso ID)
  + Algoritmas
    - BEGIN
    - i := VV(n) (pagal išorinį vardą (proceso ID) nustatomas proceso VV)
    - IF (ST[i] = BLOCKS) THEN   
       ST[i] := BLOCK   
      ELSE IF (ST[i] = READYS) THEN   
       ST[i] := READY;   
      (koreguojamas i-tojo proceso būsena)
    - IF ST[i] = READY THEN   
       PLANUOTOJAS;

(jei pakoregavus proceso būseną, procesas yra pasiruošęs, tai kviečiame planuotoją, nes galbūt, šis procesas, turės aukštesnį prioritetą už einamąjį ir procesorius bus perduotas šiam procesui)

* + - END

# Resursai

## Apie resursus

Resursas yra tai, dėl ko varžosi procesai, kitaip tariant tai yra komponentės reikalingos procesams kad įvykdytų užduotį (kodą). Dėl resursų trūkumo procesai blokuojasi, gavę reikiamą resursą, procesai tampa pasiruošusiais (jei vienintelis reikiamas resursas liko procesorius).

Resursus skirstome į tokias dvi grupes:

* **Vienkartinio naudojimo resursai** - tai tam tikra informacija, kurios laukia ir kurią gali apdoroti tik vienas procesas. Šiams resursams priklauso įvairių pavidalų informacija apie vartotojo užduotį, procesų tarpusavio pranešimai.
* **Daugkartinio naudojimo resursai** - procesorius, bendros paskirties atmintis, sisteminių pranešimų tekstai, kanalai, buferiai ir kt. Juos sistemos darbo eigoje galima naudoti kiek reikia kartų.

Procesui pasinaudojus daugkartinio naudojimo resursu - privaloma resursą atlaisvinti. Gavus vienkartinio naudojimo resursą procesai juos sunaudoja ir šių resursų atlaisvinti nereikia.



## Resursų lentelė

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resurso pavadinimas** | **Resurso grupė** | **Procesas, kuris sukuria** | **Procesas, kuris sunaikina** | **Procesas, kuris prašo** | **Procesas, kuris atlaisvina** |
| 1 kanalas | Daugkartinio | StartStop | StartStop | Input, Reader | Input, Reader |
| 2 kanalas | Daugkartinio | StartStop | StartStop | Output | Output |
| 3 kanalas | Daugkartinio | StartStop | StartStop | ProgramToExtMem, LoadProgram | ProgramToExtMem, LoadProgram |
| Input | Vienkartinio | Input | - | JobHelper | - |
| Do Input | Vienkartinio | JobHelper | - | Input | - |
| Output | Vienkartinio | Output | - | JobHelper | - |
| Do Output | Vienkartinio | JobHelper | - | Output | - |
| Programa parengta | Vienkartinio | JobHelper, LoadProgram | - | MainProc | - |
| Iškelk programą | Vienkartinio | JobHelper | - | ProgramToExtMem | - |
| Programa iškelta | Vienkartinio | ProgramToExtMem | - | JobHelper | - |
| Pertraukimas | Vienkartinio | VirtualMachine | - | Interrupt | - |
| Interrupt | Vienkartinio | Interrupt | - | JobHelper | - |
| Vartotojo atmintis | Daugkartinio | StartStop | StartStop | JobHelper | JobHelper |
| Pakrauk programą | Vienkartinio | Reader | - | LoadProgram | - |
| Laukiama | Vienkartinio | Laukimas | - | Laukimas | - |



## Paskirstytojai

Resursų paskirstytojai yra tarpinė grandis tarp procesų ir resursų. Būtent paskirstytojai organizuoja procesų darbo palaikymą, pateikdami jiems reikalingus resursus. Tai yra savarankiški programiniai fragmentai.

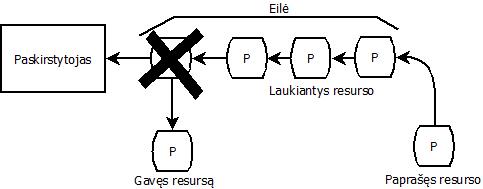
Bendru atveju kiekvienas resursas turi savo paskirstytoją. Paskirstytojas gali aptarnauti bet kokį procesą, prašantį jo valdomo resurso.

Dėl daukartinio naudojimo resursų konkuruoja procesai. Juos skirstysime prioritetinės eilės principu, tai yra – su kiekvienu resursu susiesime jo laukiančių procesų sąrašą. Tas kuris turi aukštesnį prioritetą (arba jeigu prioritetai vienodi - ankščiau paprašusio proceso), ankščiau gauna tą resursą.

Taigi pirmiausia resursą gauna aukščiausio prioriteto pirmasis eilės narys, vėliau jis iš jos išbraukiamas. Nauji procesai, kurie prašo resurso talpinami į eilę pagal prioritetą.

Paprastesnis yra vienkartinio naudojimo resursų skirstymas. Čia paskirstytojai stebi, ar neatsirado reikiamų resursų, ir jei taip, tai informuoja apie tai atitinkamą laukiantį procesą, nurodydamas resurso parametrus.

Paskirstytojo pabaigoje yra iškviečiamas planuotojas.





## Primityvai

Resursas turi keturis primityvus:

* **Kurti resursą.** Resursus kuria tik procesas. Parametrai:
  + **RS –** resurso išorinis vardas (IV)
  + **PNR –** loginis požymis (ar daukartinio naudojimo resursas)
  + **PA –** nuoroda į resurso prieinamumo aprašymą
  + **LPS –** nuoroda į LPS’ą
  + **PASK –** nuoroda į resurso paskirstytojo programą
  + **Algoritmas:**
    - BEGIN
    - r := NRVV; (naujas resurso VV)
    - RS[r] := RS; (priskiria resurso deskriptoriuje IV)
    - PNR[r] := PNR; (priskiria ar daugkartinio naudojimo resursas)
    - K[r] := \*; (resurso tėvo VV. Tai duotu metu vykstantis procesas, kuriame ir panaudojamas šis primityvas)
    - PA[r] := PA;
    - LPS[r] := LPS;
    - PASK[r] := PASK;
    - Įjungti(SR[\*], r); (į šiuo metu vykdomo proceso sukurtų resursų sąrašą įtraukiamas ir naujas sukurtas resursas)
    - END
* **Naikinti resursą.** Sunaikinti resursą gali tėvas arba jo pirmtakas. Panaikinamas resurso deskriptorius, prieš tai suaktyvinami jo laukiantys procesai. Parametrai:
  + **RS -** resurso išorinis vardas (IV)
  + **Algoritmas:**
    - BEGIN
    - r := RVV(RS); (nustatomas resurso VV pagal IV)
    - R := Pašalinti(LPS[r]); (turėsim nuorodą į pašalinamą elementą ir nuorodą į procesą)
    - WHILE R <> X (tuščiam) DO  
       IF ST[R.P] (Proceso tėvo VV) = BLOCK THEN READY  
       ELSE READYS  
       (dirbtinai atblokuoja procesą)  
       Įjungti(PPS, R.P);  
       SD[R.P] := PPS; (sūnų sąrašas)  
       R.A := “atsakymas”;  
       R := Pašalinti(LPS[r]);
    - NRP; (naikinti resursų deskriptorių)
    - PLANUOTOJAS;
    - END
* **Prašyti resurso.** Procesas, kuriam reikia resurso, iškviečia šį primityvą. Tokio proceso būsena tampa “Blokuotas”. Procesas įtraukiamas į laukiančių to resurso procesų sąrašą. Parametrai:
  + RS – resurso išorinis vardas (IV).
  + D – kokios resurso dalies prašoma.
  + A – atsakymo srities adresas į kurį pranešti
  + Algoritmas:
    - BEGIN
    - r := RVV(RS); (nustatomas resurso VV pagal IV)
    - Įjungti(LPS[r], (\*,D,A)); (procesas “įjungiamas“/įtraukiamas į laukiančių šio resurso procesų sąrašą, \* - šiuo metu vykdomas procesas)
    - PASK(r, K, L); (resurso paskirstytojo programa, K – kiek procesų aptarnauja, L – aptarnaujamų procesų VV masyvas)
    - B := true;
    - FOR J := 1 STEP 1 UNTIL K DO  
       IF L[J] <> \* THEN  
       i := L[J]  
       Įjungti(PPS, i) (procesą reikia įtraukti į pasiruošusių procesų sąrašą)  
       SD[i] := PPS  
       IF (ST[i] = BLOCK) THEN   
       ST[i] := READY  
       ELSE ST[i] := READYS   
       ELSE B:=false (reiškia tai yra einamasis procesas)  
       IF B THEN  
       ST[\*] := BLOCK  
       SD[\*] := LPS[r]  
       PROC[P[i]] := X (šį procesą vykdantį procesorių paskelbiame laisvu)  
       Pašalinti(PPS, \*)
    - PLANUOTOJAS
    - END
* **Atlaisvinti resursą.** Šį primityvą kviečia procesas, kai jis buvo gavęs pakartotinio naudojimo resursą ir kai jam jo jau nebereikia. Jis jį atlaisvina ir “Įjungia”/įtraukia į sąrašą laisvų resursų. Parametrai:
  + RS – resurso išorinis vardas (IV).
  + D – atlaisvinamos resurso dallies aprašymas
  + Algoritmas:
    - BEGIN
    - r := RVV(RS); (nustatomas resurso VV pagal IV)
    - Įjungti(PA[r], D); (resurso prieinamumo aprašymas (laisvos dalies) pakeičiamas “įjungiant“/įtraukiant atlaisvinamą resurso dalį)
    - PASK(r, K, L); (resurso paskirstytojo programa, K – kiek procesų aptarnauja, L – aptarnaujamų procesų VV masyvas)
    - FOR J := 1 STEP 1 UNTIL K DO  
       IF L[J] <> \* THEN  
       i := L[J]  
       Įjungti(PPS, i) (procesą reikia įtraukti į pasiruošusių procesų sąrašą)  
       SD[i] := PPS  
       IF (ST[i] = BLOCK) THEN   
       ST[i] := READY  
       ELSE ST[i] := READYS
    - IF K <> 0 THEN PLANUOTOJAS
    - END

# Deskriptoriai

## Procesų deskriptorius (PD)

|  |  |
| --- | --- |
| **Elemento pavadinimas** | **Aprašymas** |
| ID[i] | išorinis proceso vardas, reikalingas statiniams ryšiams tarp procesų nurodyti |
| CPU[i] | pradinė proceso būdena |
| P[i] | identifikuoja procesorių |
| OA[i] | vartotojo atminties būsena (kiek skirta vartotojo atminties resursų). |
| R[i] | nuoroda į sąrašą, kur išvardinti visi proceso gauti resursai. |
| SR[i] | proceso sukurtų resursų sąrašas (resursų deskriptorių). |
| ST[i] | proceso būsena. |
| SD[i] | nuoroda į sąrašą kuriame yra procesas (nuoroda į PPS arba LPS) |
| T[i] | nuoroda į tėvo procesą (tėvinio proc.vidinis vardas) |
| S[i] | vaikinių proc.vidinių vardų sąr. |
| PR[i] | proceso prioritetas, kuo mažesnė reikšmė, tuo procesas svarbesnis (maksimalus svarbumas = 0). |

## Procesoriaus deskriptorius (PRD)

|  |  |
| --- | --- |
| **Elemento pavadinimas** | **Aprašymas** |
| NP | procesų skaičius |
| PROC[i] | i-tojo proceso Vidinis Vardas (VV). |

## Resursų deskriptorius (RD)

|  |  |
| --- | --- |
| **Elemento pavadinimas** | **Aprašymas** |
| RID[i] | resurso ID (išorinis vardas) |
| PNR[i] | ar tai pakartotinio panaudojimo resursas |
| K[i] | vidinis vardas resurso kūrėjo (proceso) |
| PA[i] | resurso prieinamumo aprašymas (laisvos dalies) nuoroda į sąrašo pražią. |
| LPS[i] | nuoroda į LPS praždią |
| PASK[i] | resurso paskirstytojo programos adresas |
| APSKAITOS INFORMACIJA | Į resurso deskriptorių įtraukiama pagalbinė apskaitos informacija. |



# Procesų paketas

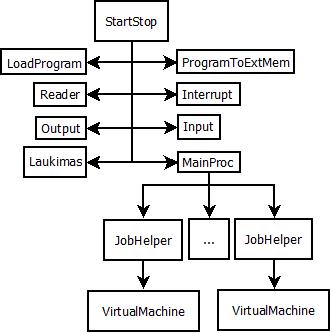


## Apžvalga

Naudosime šiuos procesus:

* **StartStop** – šakninis procesas, sukuriantis bei naikinantis sisteminius procesus ir resursus.
* **ProgramToExtMem** – užduotis perkelia vartotojo programą iš vartotojo atminties į išorinę atmintį.
* **LoadProgram** – iš išorinės atminties duomenys perkeliami į vartotojo atmintį.
* **Input**– procesas darbui su įvedimu.
* **Output** – procesas darbui su išvedimu.
* **MainProc** – procesas, valdantis JobHelper procesus.
* **JobHelper** – virtualios mašinos proceso tėvas, valdantis virtualios mašinos proceso darbą.
* **VirtualMachine** – procesas, atsakantis už vartotojiškos programos vykdymą.
* **Interrupt** – procesas, apdorojantis virtualios mašinos pertraukimą sukėlusią situaciją.

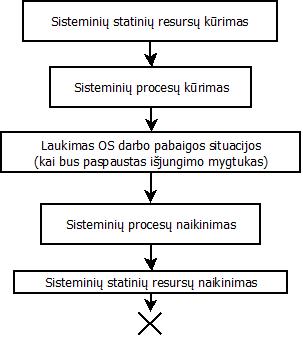
Beveik visi procesai yra sukuriami sistemos darbo pradžioje proceso StartStop. StartStop nekuria tik 2 procesų – JobHelper (kiekvienai naujai vartotojo užduočiai MainProc kuria po naują procesą JobHelper) ir VirtualMachine, kurį kuria JobHelper. Bendrą vaizdą galėtume pavaizduoti tokia schema:





## StartStop

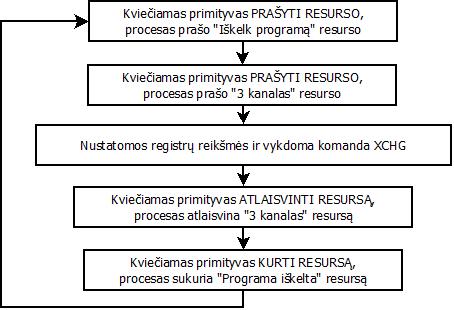
**StartStop** – šakninis procesas, įjugus kompiuterį pasileidžia automatiškai. Kuria kitus sisteminius procesus bei resursus ir laukia OS darbo pabaigos (kol bus paspaustas išjungimo mygtukas).





## ProgramToExtMem

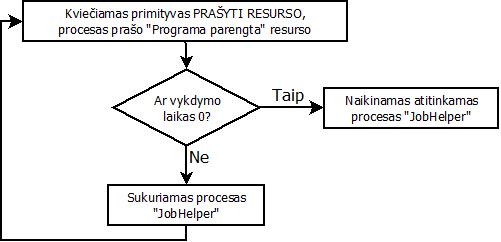
**ProgramToExtMem** – sisteminis procesas, perkeliantis duomenis iš vartotojo atminties į išorinę.





## MainProc

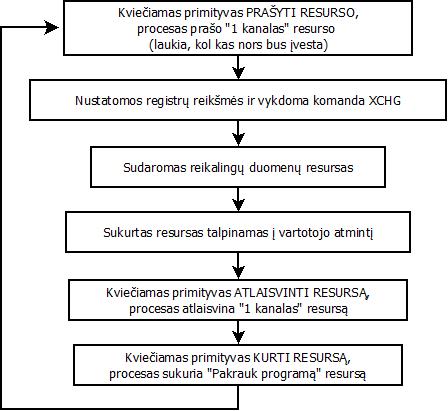
**MainProc** – procesas, kuriantis ir naikinantis procesus „JobHelper“.





## Reader

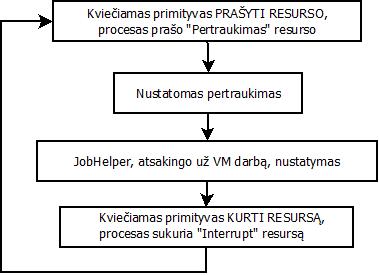
**Reader** - proceso paskirtis yra paimti išorinės atminties bloko numerį, kuriame yra programa, kuri bus užkraunama.





## Interrupt

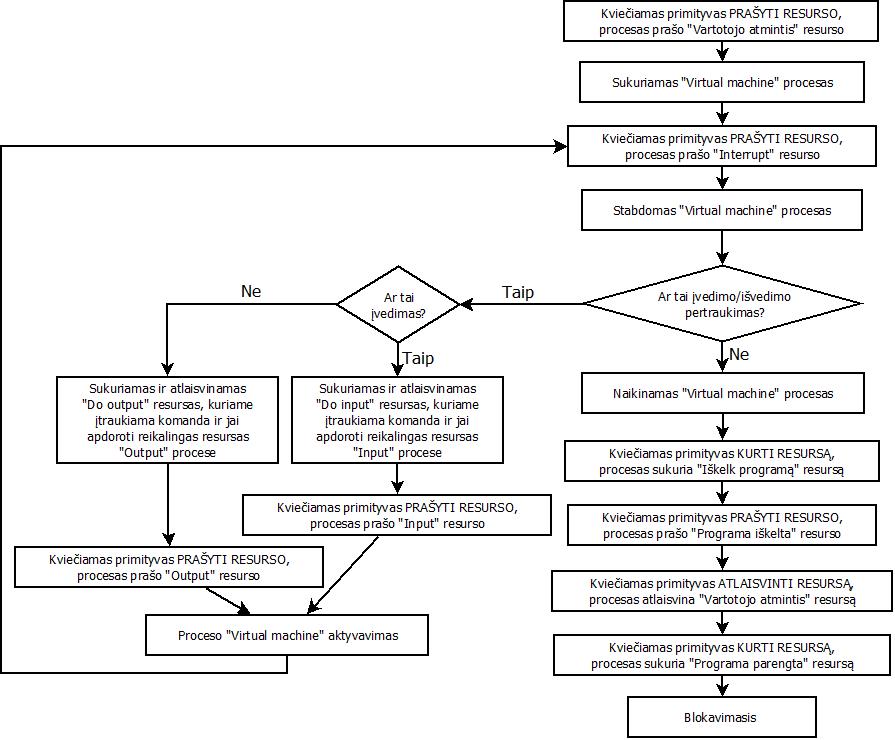
**Interupt** – procesas, skirtas apdoroti kylančius pertraukimus.





## JobHelper

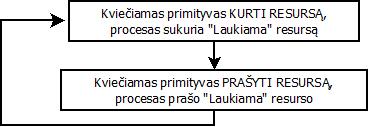
**JobHelper** - procesas valdo vieną procesą „VirtualMachine“.





## Laukimas

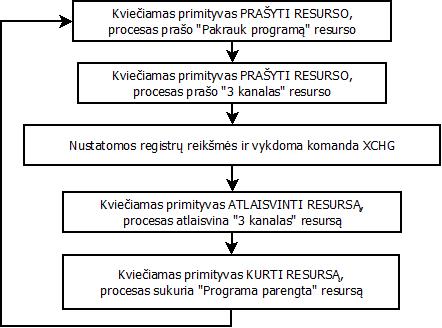
**Laukimas** - procesas su mažiausiu įmanomu prioritetu. Jis tiesiog užims procesoriaus laiką, kol atsiras kitas pasiruošęs procesas.





## LoadProgram

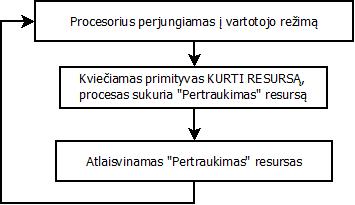
**LoadProgram** - proceso paskirtis yra išorinėje atmintyje esančią programą perkelti į vartotojo atmintį.

****



## VirtualMachine

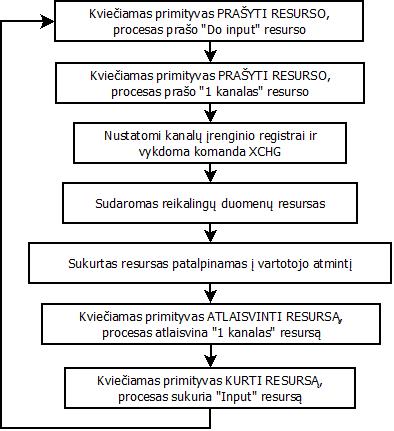
**VirtualMachine** – proceso paskirtis yra vykdyti vartotojo programas. Jį kuria ir naikina procesas JobHelper. Jų yra tiek kiek yra JobHelper procesų.





## Input

**Input** – procesas, skirtas darbui su įvedimu.





## Output

**Output** – procesas, skirtas darbui su išvedimu.

