# Procesai

Arnas Steponavičius

Arnas Radzevičius

Juozas Natkevičius

Info 2 gr.

VU MIF

2019

Multiprograminės operacinės sistemos (MOS)

projektas

Procesas – tai vykdoma programa, kartu su esamomis registrų reikšmėmis ir savo kintamaisiais. Kiekvienas procesas turi savo virtualų procesorių. Procesas – tai kokioje nors veiklumo stadijoje esanti programa. Tuo tarpu programa – tai tik tam tikras baitų rinkinys. Veiklumo stadiją apibūdina proceso aprašas. Apraše ir yra laikomi visi procesui reikalingi parametrai, tokie kaip virtualaus procesoriaus registrų reikšmės ar jam reikalingi kintamieji.

Procesų aprašai – dinaminiai objektai. Tai reiškia, kad jie gali būti sukurti ar sunaikinti jau sistemos veikimo metu. Procesus kuria procesai, iš to seka, kad turi būti vienas pagrindinis procesas, kuris sukurs visus kitus. Paprastai procesus galima suskirstyti į vartotojiškus ir sisteminius.

Sisteminių procesų paskirtis – aptarnauti vartotojiškus. Sisteminiai procesai yra kuriami paleidžiant sistemą, o naikinami – naikinant sistemą. Taigi visą sistemos gyvavimo laiką jie yra pasiruošę dirbti jiems skirtą darbą. Paprastai paleidus sistemą, visi sisteminiai procesai anksčiau ar vėliau užsiblokuoja. Bet tai nereiškia, kad jie yra neveiklūs. Jie laukia, kol jie galės atlikti savo darbą, kuris paprastai būna susijęs su vartotojiška užduotimi.Pavyzdžiui, atėjus signalui iš vartotojo sąsajos apie naują užduotį, yra atlaisvinamas tam tikras resurso elementas. Jo dėka atsiblokuoja vienas iš sisteminių procesų, kuris nuveikia tam tikrą darbą ir atlaisvina vieną ar kelis resursus. Taigi, sisteminiai procesai vienas po kito atsiblokuoja, kol galų gale sukuriama vartotojo užduotis. Vartotojo užduočiai baigus darbą, neužsiblokavę procesai ne už ilgo užsiblokuoja, ir vėl prasideda laukimas.

Tuo tarpu vartotojiško proceso paskirtis yra vykdyti vartotojo programą. Vartotojiški procesai yra sukuriami sisteminių procesų jau veikiant sistemai. Kartu su vartotojišku procesu gali būti sukurti vienas ar keli sisteminiai procesai, skirti aptarnauti vartotojišką procesą.

Proceso aprašas TProcess:

* FList: TProcessList; - procesų sąrašas, kuriam priklauso procesas. Tai gali būti pasiruošusių procesų s ąrašas arba blokuotų, dėl kokio nors resurso, procesų sąrašas.
* FID: Integer;- Vidinis proceso vardas. Kiekvienas sisteminis objektas turi savo unikalų numerį.
* SavedRegisters: TSavedRegisters; - Proceso išsaugota procesoriaus būsena. Naudojama perduodant valdymą procesui.
* FProcessor: TProcessor; nuoroda į procesorių.
* FCreatedRes: TResourceList; - Proceso sukurti resursai.
* FOwnedRes: TElementList; - Procesui kūrimo metu perduoti resursai.
* FState: TProcessState; - Proceso būsena.
* FPriority: Integer; - Proceso prioritetas. Pasirinkta sistema: kuo didesnė reikšmė, tuo procesas laikomas svarbesniu .
* FParent: TProcess; - nuoroda į procesą–tėvą.
* FChildren: TProcessList; - procesų–vaikų sąrašas.
* FUserName: AnsiString; - Išorinis vardas. Naudojamas patogesniam proceso identifikavimui OS stebėjimo metu.
* FStack: Pointer; - proceso aplinka – stekas. Kiekvieną kartą perduodant valdymą procesui yra gražinama jo aplinka. Kiekvieną kartą, iš proceso atimant procesorių, jo aplinka yra išsaugoma.
* FStackSize: Integer; - proceso aplinkos – steko dydis.

Procesų sąrašo aprašas TProcessList:

* FProcesses: array[ ] of TProcess: - Dinaminis procesų sąrašas.
* FCount: Integer ; Procesų skaičius procesų masyve.

### Procesų būsenos

Galime išskirti procesų būsenas:

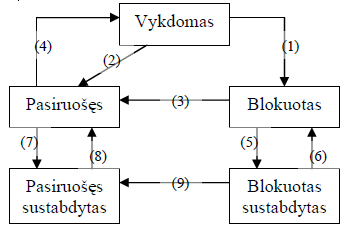
* Vykdomas - turi procesorių.

• Blokuotas - prašo resurso (išskyrus procesorių).

• Pasiruošęs – vienintelis trūkstamas resursas yra procesorius.

• Sustabdytas – kito proceso sustabdytas procesas.

Procesoriaus resursas yra reikalingas visiems procesams. Nė vienas procesas netaps vykdomu (taigi, ir nedirbs savo darbo), neturėdamas procesoriaus.



Diagramoje viršuje vaizduojami šie procesų būsenų perėjimai:

1. Vykdomas procesas blokuojasi jam prašant ir negavus resurso.

2. Vykdomas procesas tampa pasiruošusiu atėmus iš jo procesorių dėl kokios nors priežasties (išskyrus resurso negavimą).

3. Blokuotas procesas tampa pasiruošusiu, kai yra suteikiamas reikalingas resursas.

4. Pasiruošę procesai varžosi dėl procesoriaus. Gavęs procesorių procesas tampa vykdomu.

5. Procesas gali tapti sustabdytu blokuotu, jei einamasis procesas jį sustabdo, kai jis jau ir taip yra blokuotas.

6. Procesas tampa blokuotu iš blokuoto sustabdyto, jei einamasis procesas nuima būseną sustabdytas.

7. Procesas gali tapti pasiruošusiu sustabdytu, jei einamasis procesas jį sustabdo, kai jis yra pasiruošęs.

8. Procesas tampa pasiruošusiu iš pasiruošusio sustabdyto, jei einamasis procesas nuima būseną sustabdytas

9. Procesas tampa pasiruošusiu sustabdytu iš blokuoto sustabdyto, jei procesui yra suteikiamas jam reikalingas resursas.

## Planuotojas

Planuotojas tai procesoriaus resurso paskirstytojas. Planuotojo paskirtis yra paimti procesorių iš proceso, peržvelgti pasiruošusių procesų sąrašą, išrinkti pasiruošusį procesą, kuris planuotojo manymu yra tinkamiausias, ir perduoti procesorių jam. Planuotojo tikslai:

• Užtikrinti, kad kiekvienas procesas pagristą laiko tarpą (nei per daug, nei per mažai) turėtų procesorių;

• Užtikrinti maksimalų procesoriaus užimtumą;

• Sumažinti iki minimumo atsakymo laiką vartotojams;

• Siekti kuo didesnio efektyvumo.

Planuotojo veiksmų seką galima būtų pavaizduoti tokia diagrama:

Einamasis

procesas

blokuotas?

Einamasis

procesas įtraukiamas į

blokuotų procesų sąrašą

Yra

pasiruošusių

procesų?

Aukščiausio prioriteto procesui

suteikiamas valdymas

Vykdomas procesas, kuriam ką

tik suteikėme valdymą

Taip

Taip

Ne

Mūsų planuotojas remsis procesų prioritetais. Proceso prioritetas – tai proceso svarba, įvertinta kokioje nors skalėje, mūsų atveju tai bus nuo 0 iki 4, kur 4 yra maksimalus svarbumas. Procesas, turintis didesnį prioritetą, atsiduria arčiau procesų sąrašo pradžios. Visi procesų sąrašai yra rūšiuojami pagal proceso prioritetą, ir arčiau pradžios esantis procesas turi didesnę galimybę tapti vykdomu (pasiruošusių procesų sąrašo atveju) arba pasiruošusiu (blokuotų procesų sąrašo atveju). Sisteminių procesų paskirtis – aptarnauti vartotojiškus, todėl paprastai vartotojiški procesai turi mažesnį prioritetą nei sisteminiai:

|  |  |
| --- | --- |
| **Procesų prioritetai** | |
| Proceso prioritetas | Proceso(-ų) pavadinimas(-ai) |
| 4 | StartStop |
| 3 | Interrupt; Channel Interrupt, LoadProgram; JCL; Loader; MainProc, PutLine |
| 2 | JobHelper |
| 1 | VirtualMachine |
| 0 | Idle |

## Primityvai

Procesų primityvų paskirtis – pateikti vienodą ir paprastą vartotojo sąsają darbui su procesais. Darbui su procesais skirti 4 primityvai:

1. **Kurti procesą**. Šiam primityvui perduodama nuoroda į jo tėvą, jo pradinė būsena, prioritetas, perduodamų elementų sąrašas ir išorinis vardas. Pačio primityvo viduje vyksta proceso kuriamasis darbas. Jis yra registruojamas bendrame procesų sąraše, tėvo-sūnų sąraše, skaičiuojamas vidinis identifikacijos numeris, sukuriamas jo vaikų procesų sąrašas (tuščias), sukurtų resursų sąrašas ir t.t.

2. **Naikinti procesą**. Pradedama naikinti proceso sukurtus resursus ir vaikus. Vėliau išmetamas iš tėvo sukurtų procesų sąrašo. Toliau išmetamas iš bendro procesų sąrašo ir, jei reikia, iš pasiruošusių procesų sąrašo. Galiausiai naikinami visi jam perduoti resursai ir proceso deskriptorius yra sunaikinamas.

3. **Stabdyti procesą**. Keičiama proceso būsena iš blokuotos į blokuotą sustabdytą arba iš pasiruošusios į pasiruošusią sustabdytą. Einamasis procesas stabdomas tampa pasiruošusiu sustabdytu.

4. **Aktyvuoti procesą.** Keičiama proceso būsena iš blokuotos sustabdytos į blokuotą, ar pasiruošusios sustabdytos į pasiruošusią.

Kiekvieno primityvo programos gale yra kviečiamas planuotojas.

# Resursai

Resursas yra tai, dėl ko varžosi procesai. Dėl resursų trūkumo procesai blokuojasi, gavę reikiamą resursą, procesai tampa pasiruošusiais. Resursus galima skirstyti į:

• Statinius resursus. Kuriami sistemos kūrimo metu. Tai mašinos resursai, tokie kaip procesorius, atmintis ar kiti resursai, kurie sistemos veikimo metu nėra naikinami. Šie resursai gali būti laisvi, kai nė vienas procesas jų nenaudoja, arba ne, kada juos naudoja vienas ar keli, jei tą resursą galima skaldyti, procesai.

• Dinaminius resursus. Kuriami ir naikinami sistemos darbo metu. Šie resursai naudojami kaip pranešimai. Kartu su jais gali ateiti naudinga informacija. Kartais šio tipo resursas pats yra pranešimas.

Prašyti resurso – tai prašyti resurso elemento. Kiekvienas resursas turi laukiančių procesų sąrašą (jis gali būti ir tuščias). Kiekvienas procesas prašęs, ir negavęs resurso elemento, yra ne tik užblokuojamas, bet ir įdedamas į resurso laukiančių procesų sąrašą.

Resurso aprašas TResource:

* FID: Integer ; - vidinis resurso vardas. Kiekvienas sisteminis objektas turi savo unikalų numerį.
* FCreator: TProcess; -Nuoroda į procesą, sukūrusį šį resursą.
* FList: TElementList; - Nuoroda į resurso elementų sąrašą.
* FWaitingProc : TProcessList; -Nuoroda į šio resurso laukiančių procesų sąrašą.
* FWaitingCount: TIntegerList; -Nuoroda į šio resurso laukiančių procesų paprašytų resurso kiekių sąrašą.
* FResourceList: TResourceList;- nuoroda į visų resursų sąrašą

Resursų sąrašo aprašas TResourceList

* Count: Integer ; - sąraše esančių resursų kiekis.
* Resources: array[ ] of TResource ; - dinaminis masyvas, kurio elementai – resursai.

Resurso elemento aprašas TResElement

* FElementList: TelementList- nuoroda į resurso elementų sąrašą, kuriame yra šis resurso elementas.
* FReceiver: TProcess;- procesas, kuris turi gauti šį resurso elementą. Jei šio lauko reikšmė lygi nil, tai laikoma, kad šį elementą gali gauti bet kuris procesas.
* FSender: TProcess; - procesas, atlaisvinęs šį resurso elementą

Resurso elementų sąrašas TElementList

* FResource: TResource; - Nuoroda į resursą, kuriam priklauso šis elementų sąrašas
* FCount: Integer; - Elementų skaičius sąraše. Šis skaičius nurodo, kiek resurso elementų yra laisvų (t.y. prieinamų procesams) iš viso.
* FElement:array [ ] of TResElement; - dinaminis elementų sąrašas. Kreipiantis su tam tikru indeksu pasiekiamas atitinkamas elementas.

## Resurso primityvai

Resursas turi keturis primityvus:

1. **Kurti resursą**. Resursus kuria tik procesas. Resurso kūrimo metu perduodami kaip parametrai: nuoroda į proceso kūrėją, resurso išorinis vardas. Resursas kūrimo metu yra: pridedamas prie bendro resursų sąrašo, pridedamas prie tėvo suskurtų resursų sąrašo, jam priskiriamas unikalus vidinis vardas, sukuriamas resurso elementų sąrašas ir sukuriamas laukiančių procesų sąrašas.

2. **Naikinti resursą**. Resurso aprašas išmetamas iš jo tėvo sukurtų resursų sąrašo, naikinamas jo elementų sąrašas, atblokuojami procesai, laukiantys šio resurso, išmetamas iš bendro resursų sąrašo, ir, galiausiai naikinamas pats aprašas.

3. **Prašyti resurso.** Šį primityvą kartu su primityvu “atlaisvinti resursą” procesai naudoja labai dažnai. Procesas, iškvietęs šį primityvą, yra užblokuojamas ir įtraukiamas į to resurso laukiančių procesų sąrašą. Sekantis šio primityvo žingsnis yra kviesti resurso paskirstytoją.

4. **Atlaisvinti resursą**. Šį primityvą kviečia procesas, kuris nori atlaisvinti jam nereikalingą resursą arba tiesiog perduoti pranešimą ar informaciją kitam procesui. Resurso elementas, primityvui perduotas kaip funkcijos parametras, yra pridedamas prie resurso elementų sąrašo. Šio primityvo pabaigoje yra kviečiamas resursų paskirstytojas.

## Resurso paskirstytojas

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

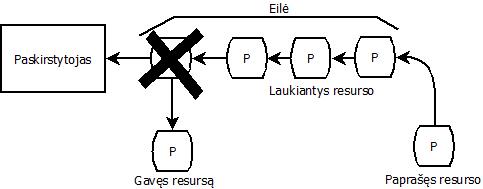
Bendru atveju kiekvienas resursas turi savo paskirstytoją. Paskirstytojas gali aptarnauti bet kokį procesą, prašantį jo valdomo resurso.

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.



# Procesų paketas

Naudosime šiuos procesus:

|  |  |  |  |
| --- | --- | --- | --- |
| **Pavadinimas** | **Kas sukuria/ sunaikiną procesą** | **Paskirtis** | **Su kokiais resursais dirbama** |
| StartStop | Sukuriamas ir naikinamas  automatiškai kartu su sistema | Sukurti sisteminius procesus bei resursus, pasirūpinti korektiška sistemos darbo pradžia ir pabaiga. | Sisteminiai statiniai |
| LoadProgram | StartStop | Užduoties nuskaitymui iš išorinės atminties. | “Failo Pavadinimas”  Supervizorinė atmintis  “Užduotis supervizorinėje atmintyje” |
| JCL | StartStop | suskaidyti užduoties programą blokais ir organizuoti juos kaip resursus. | Sisteminiai statiniai  “Užduotis supervizorinėje atmintyje”  “Eilutė atmintyje”  “Užduoties programa supervizorinėje atmintyje” |
| Loader | StartStop | Perkelti duomenis iš išorinės atminties į vartotojo atmintį; | Sisteminiai statiniai  Supervizorinė atmintis  Vartotojo atmintis  „Duomenų užkrovimo įvykis“  „Loader darbo pabaiga“ |
| MainProc | StartStop | valdyti JobHelper procesus | Sisteminiai statiniai  „MainProc užduotis“ |
| JobHelper | MainProc | virtualios mašinos proceso tėvas. valdyti virtualios mašinos proceso darbą | Sisteminiai statiniai  “Loader darbo pabaiga”  Supervizorinė atmintis  “Interrupt pranešimas”  „Vartotojo atmintis“  „MainProc užduotis“  „Duomenų užkrovimo įvykis“  „Vartotojo įvedimas“ |
| VirtualMachine | JobHelper | atsakyti už vartotojiškos programos vykdymą | Sisteminiai statiniai  “Pertraukimas” |
| Interrupt | StartStop | apdoroti virtualios mašinos pertraukimą sukėlusią situaciją | Sisteminiai statiniai  “Pertraukimas”  “Interrupt pranešimas” |
| Channel Interrupt | StartStop | Apdoroti kanalų įrenginio pertraukimus | „Vartotojo įvedimas“, „Failo Pavadinimas“, „POS pabaiga“, „Pertraukimo įvykis“, „Kanalų įrenginio interrupt“ |
| PutLine | StartStop | Persiųsti į išvedimo įrenginį eilutę (spausdintuvą) iš supervizorinės atminties | “Eilutė atmintyje”  „Kanalų įrenginys išvedimas“ |
| Idle | StartStop |  | „Kanalų įrenginio interrupt“ |

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:

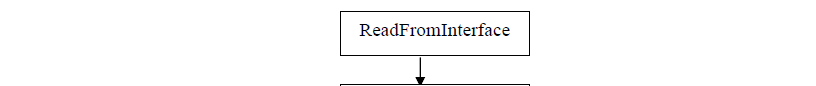
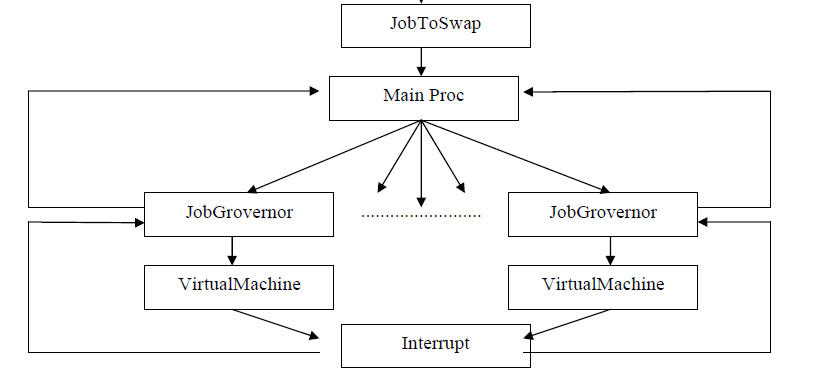
LoadProgram

JCLam

Loaderam

JobHelper

JobHelper



## StartStop

Šis procesas atsakingas už sistemos darbo pradžią ir pabaigą. Įjungus kompiuterį šis procesas pasileidžia automatiškai. Šio proceso paskirtis – sisteminių procesų ir resursų kūrimas. Šio proceso schema atrodo taip:

**START**

**STOP**

Sisteminių procesų inicializacija

Blokavimasis laukiant „POS pabaiga“ resurso

Sisteminių procesų naikinimas

Sisteminių resursų naikinimas

Sisteminių resursų inicializacija

## LoadProgram

Šį procesą kuria ir naikina procesas StartStop. Proceso paskirtis yra išorinėje atmintyje esančią programą perkelti į supervizorinę atmintį. Šio proceso schema atrodo taip:

Blokavimasis laukiant „Failo Pavadinimas“ resurso.

Įvedimo supervizorinėje atmintyje validavimas

Failo nuskaitymas ir suskaidymas blokais.

Blokavimasis laukiant supervizorinės atminties resurso.

Atlaisvinamas resursas „Užduotis supervizorinėje atmintyje“

“.

Blokų kopijavimas į supervizorinę atmintį.

JCL

Procesą JCL (Job Control Language) kuria ir naikina procesas StartStop. Proceso JCL paskirtis – gautus blokus iš proceso LoadProgram suskirstyti į programos blokus, ir atidavus procesui MainProc, vėl blokuotis laukiant pranešimo iš LoadProgram.

Žodis prijungiamas prie programos sąrašo

Imamas sekantis žodis iš supervizorinės atminties

TAIP

Blokavimasis laukiant resurso „Užduotis supervizorinėje atmintyje“

Programos blokų sąrašo inicijavimas

Imamas pirmas žodis iš supervizorinės atminties

Ar žodis yra „$HDR“?

Atlaisvinamas resursas „Eilutė atmintyje“, su parametrais: „Nėra vartotojo programos”

Ar žodis yra „$BDY“?

NE

TAIP

Ar tai žodis „$END“ arba tuščia?

NE

Atlaisvinamas resursas „Eilutė atmintyje“, su parametrais: „Nėra programos antraštės”

Imamas sekantis žodis iš supervizorinės atminties

Ar žodis yra „$BDY“?

Žodis prijungiamas prie programos sąrašo

TAIP

NE

Imamas sekantis žodis iš supervizorinės atminties

Žodis prijungiamas prie programos sąrašo

TAIP

Ar tai žodis „$END“?

NE

Atlaisvinamas resursas „MainProc užduotis“ su pranešimu „Vykdymo laikas = 1“

Atlaisvinti „Eilutė atmintyje“ resursą su parametrais: „Nėra programos pabaigos žymės“

NE

Ar žodis tuščias?

NE

TAIP

TAIP

## Loader

Procesą kuria ir naikina procesas StartStop. Šio proceso paskirtis – supervizorinėje atmintyje esančius blokus perkelti į vartotojo atmintį.

Blokavimasis laukiant „Supervizorinė atmintis“ resurso

Blokavimasis laukiant resurso „Vartotojo atmintis“

Įvykdoma komanda XCHG

Atlaisvinamas resursas „Supervizorinė atmintis“

Atlaisvinamas resursas “Loader darbo pabaiga”

Blokavimasis laukiant „Duomenų užkrovimo įvykis“ resurso

## MainProc

Šio proceso paskirtis – kurti ir naikinti procesus JobHelper. Procesas prašo užduoties, kurią reikia įvykdyti. Tikrinama ar resurso vykdymo laikas nelygus 0, t.y. ar užduotis nevykdoma per ilgai. Jei laikas nelygus nuliui - kuriamas procesas JobHelper priešingu atveju - naikinamas procesas. JobHelper kuris atsiuntė pradinį resursą su nuliniu vykdymo laiku. Atlikęs savo darbą procesas MainProc vėl blokuojasi laukdamas resurso.

TAIP

NE

Kuriamas procesas JobHelper

Blokavimasis laukiant „MainProc užduotis“ resurso

Naikinamas procesas JobHelper

Vykdymo laikas=0?

## JobHelper

Procesą (galima sakyti ir procesus – jų vienu metu gali būti keli) JobGorvernor kuria procesas MainProc. Proceso JobHelper paskirtis – kurti, naikinti ir padėti procesui VirtualMachine atlikti savo darbą. “Padėti – tai atlikti veiksmus, kurių VirtualMachine, procesoriui dirbant vartotojo režimu, nesugeba atlikti. Vienas JobHelper aptarnauja vieną virtualią mašiną. Detali šio proceso schema yra 9-oje diagramoje.

Atlaisvinamas resursas „Duomenų užkrovimo įvykis“

Blokavimasis laukiant resurso „Loader darbo pabaiga“

Blokavimasis laukiant „Supervizorinė atmintis“ resurso – kuriama puslapių lentelė

Proceso „Virtual Machine“ kūrimas

Blokavimasis laukiant proceso „Interrupt“ pranešimo

Ar I/O pertraukymas?

Proceso „Virtual Machine“ stabdymas

NE

TAIP

Naikinti procesą „Virtual Machine“

Atlaisvinti „Vartotojo atmintis“ resursą

Blokavimasis laukiant „Vartotojo įvedimas“ resurso

Ar GET pertraukimas?

Į išvedimo srautą pasiunčiama atsiųstu adresu esantis vartotojo atminties blokas



NE

Proceso „VirtualMachine“ aktyvavimasis

Blokavimasis laukiant „Vartotojo atmintis” resurso

Atlaisvinti re sursą „MainProc užduotis “ su pranešimu „Vykdymo laikas = 0”

## VirtualMachine

Procesą kuria ir naikina procesas JobHelper. Proceso VirtualMachine paskirtis yra vykdyti vartotojo užduoties programą. Šių procesų yra tiek kiek yra procesų JobHelper. Persijungiama į vartotojo režimą. Interpretuojama programa kol neįvyksta pertraukimas. Įvykus pertraukimui virtuali mašina išsaugo savo procesoriaus būseną, valdymas perduodamas pertraukimą apdorosiančioms programoms. Kuriamas resursas „Pertraukimas“ skirtas procesui Interrupt, kuris identifikuos pertraukimą ir perduos informaciją procesui JobHelper.

Procesoriaus perjungimas į vartotojo režimą.

Vykdoma vartotojo programa.

Sukuriamas resursas „Pertraukimas“ su pranešimu „HALT“.

## Interrupt

Šio proceso paskirtis – reaguoti į pertraukimus, kilusius virtualios mašinos darbo metu. Pradžioje laukiama resurso „Pertraukimas“. Procesas nustato pertraukimo tipą apklausinėdamas programų nustatytas sisteminių kintamųjų reikšmes. Galiausiai kuriamas resursas „Interrupt“, skirtas procesui JobHelper.

Blokavimasis laukiant resurso „pertraukimo įvykis“

Pertraukimo įvykio identifikavimas

Pertrauktosios VM tėvinio proceso JobHelper identifikavimas

Pranešimo apie pertraukimą procesui JobHelper sukūrimas

## ChannelInterrupt

Procesą kuria ir naikina StartStop. Paskirtis – apdoroti į įvedimo įrenginį įvestus duomenis.

TAIP

Blokavimasis laukiant „kanalo įrenginio interrupt“ resurso

Atlaisvinamas „Failo Pavadinimas“ resursas.

Ar įvestas „Enter“?

Ar įvestas „Esc“?

Atlaisvinamas „POS pabaiga“ resursas.

Kanalų įrenginio pertraukimo identifikavimas

Tikrinama PTR registro reikšmė

Tikrinama PTR registro reikšmė

Ar PTR reikšmė 0?

NE

Atlaisvinamas „Vartotojo įvedimas“ resursas.

Ar PTR reikšmė 0?

TAIP

Atlaisvinamas resursas „pertraukimo įvykis“ su pranešimu „HALT“

NE

NE

TAIP

## PutLine

Procesą kuria ir naikina StartStop. Šio proceso paskirtis – į išvedimo srautą pasiųsti kokioje nors atmintyje esančius duomenis. Procesas gavęs resursą „Eilutė atmintyje“ užsiblokuoja kol atsilaisvins kanalų įrenginys, jam atlaisvinus įvykdoma komanda XCHG, po kurios atlaisvinamas resursas „Kanalų įrenginys“. Rezultate, eilutė, esanti atmintyje, bus išvesta į išvedimo lauką vartotojui.

Blokavimasis laukiant resurso „Eilutė atmintyje“

Blokavimasis laukiant „kanalų įrenginys išvedimas“ resurso

Eilutės išvedimas

„kanalų įrenginys išvedimas“ resurso atlaisvinimas

## Idle

Procesą kuria ir naikina StartStop. Paskirtis - palaikyti procesoriaus užimtumą ir tikrinti kanalų įrenginio registrą CH1 (įvedimas). Kuomet CH1 reikšmė yra ne nulis, atlaisvinamas kanalų įrenginio interrupt resursas su pranešimu „CH1 reikšmė”.

NE

Tikrinam kanalo įrenginio registro CH1 reikšmę

Atlaisvinamas „kanalų įrenginio interrupt“ resursas su pranešimu „CH1 reikšmė“

Ar reikšmė=0?