|  |
| --- |
|  |
| **Operacinės sistemos**  Trečiasis praktinis darbas  „Multiprograminės operacinės sistemos projektas“ |
| Darbą atliko:  Inf 4 kurso studentas Lukas Chodosevičius |

Turinys

[I. Realios mašinos aprašas 3](#_Toc475271610)

[ Realios mašinos komponentu schema ir jų tarpusavio sąryšis: 3](#_Toc475271611)

[ Realią mašiną sudaro: 3](#_Toc475271612)

[ Detalesnė kievkieno komponento aprašas: 3](#_Toc475271613)

[II. Virtualios mašinos aprašas 4](#_Toc475271614)

[ Procesorius 4](#_Toc475271615)

[ Atmintis 4](#_Toc475271616)

[ Virtaulios mašinos instrukcijos 4](#_Toc475271617)

[ Virtualios mašinos interpretuojamo failo formatas 5](#_Toc475271618)

[ Virtualios mašinos bendravimo su įvedimo/išvedimo įrenginiais aprašymas 5](#_Toc475271619)

[III. Virtuali mašina visos operacinės sistemos kontekste 5](#_Toc475271620)

# Operacinės sistemos modelis

## Procesai

Procesas – tai vykdoma programa, kartu su esamomis registrų reikšmėmis ir savo kintamaisiais. Nors skirtumas tarp programos ir proceso nėra didelis, bet jis svarbus. Procesas – tai kokioje nors veiklumo stadijoje esanti programa. Tuo tarpu programa – tai tik tam tikras baitų rinkinys.

Paprastai procesus galima suskirstyti į vartotojiškus ir sisteminius. Sisteminių procesų paskirtis – aptarnauti vartotojiškus. Tuo tarpu vartotojiško proceso paskirtis yra vykdyti vartotojo programą. Kiek vėliau apie procesus bus detaliau.

### Struktūra

**Fid** – unikalus identifikatorius.

**Name** – proceso vardas.

**State** – proceso būsena. Galimos: **kProcessStateRunning**, **kProcessStateReady**, **kProcessStateBlocked**, **kProcessStateReadyStoped**, **kProcessStateBlocketStoped**.

**Name** – proceso vardas.

**PriorityState** – proceso svarba, galimos reikšmės: **kProcessPriorityLow**, **kProcessPriorityMedium**, **kProcessPriorityHigh**.

**PriorityCalculator** – apsikaičiojamas proceso prioritetas. Priklauso nuo **PriorityState**.

**Parent** – iš kurio proceso sukurtas

**Children** – procesai kurie yra sukurti iš šio proceso.

**CreatedResources** – resursai sukurti šio proceso.

**OwnerResourceElements** – resursų elementai, kuriuos pasisavino procesas.

### Primityvai

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

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

### 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.

* **kProcessStateRunning** – procesas šiuo metu turi procesoriu ir atliekamos jo darbas.
* **kProcessStateReady** – procesas šiuo metu laukia kada atsilaisvins procesorius.
* **kProcessStateBlocked** – procesas yra užblokuotas nes jam reikia tam tikro resurso. Vėliau bus aptarti resursai.
* **kProcessStateReadyStoped** – tas pats kaip **kProcessStateReady**, tačiau procesas yra sustabdytas ir nepasieks **kProcessStateRunning** tol kol nebus pratęstas.
* **kProcessStateBlocketStoped** – tas pats kaip **kProcessStateBlocked**, tačiau procesas yra sustabdytas ir nepasieks **kProcessStateReady** tol kol nebus pratęstas.

### Planuotojas

Trumpai kalbant, planuotojo paskirtis yra atimti procesorių iš proceso, peržvelgti pasiruošusių procesų sąrašą, išrinkti pasiruošusį procesą, kuris planuotojo manymu yra tinkamiausias, ir perduoti procesorių jam. Tinkamiausias procesas yra išrenkamas su didžiausia prioriteto reikšmė.

### Prioritetas

Kiekvienas procesas turi savo prioritetą kuris yra apsikaičiojamas **PriorityCalculator**, tai vėliau panaudoja **Planuotojas.**

**PriorityCalculator** – tai grubiai tariant paprastas algoritmas, kurio tikslas:

* Išskirstyti procesų darbą, kad būtu kuo mažesni bendras procesų laukimas.
* Skirstant procesus atsižvelgti **PriorityState**.

Šiame projekte bus naudojamas šis algoritmas apskaičiuoti proceso prioritetui:

ProcessPriority = (**kProcessPriorityLow** \* 50 + **kProcessPriorityMedium** \* 75 + **kProcessPriorityHigh** \* 100) + (**kProcessPriorityLow** \* 5 + **kProcessPriorityMedium** \* 7 + **kProcessPriorityHigh** \* 9) \* NumberOfTimesWasNotSelectedReady – NumberOfTimesRunning.

Trumpas paaiškinimas:

* **kProcessPriorityLow, kProcessPriorityMedium, kProcessPriorityHigh** – įgauna 1 arba 0 priklauso nuo **PriorityState**.
* NumberOfTimesWasNotSelectedReady – kiek kartu procesas nebuvo išrinktas būti paleistu.
* NumberOfTimesRunning – atvirkščiai NumberOfTimesWasNotSelectedReady.

## Resursai

Resursas - 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.
* Dinaminius resursus. Kuriami ir naikinami sistemos darbo metu. Šie resursai naudojami kaip pranešimai. Kartu su jais gali ateiti naudinga informacija.

### Struktūra

**Fid** – unikalus identifikatorius.

**Name** – resurso vardas.

**WaitingPeocesses** – procesai kurie laukio iš šio resurso elementų.

**WaitingCount** – elementų skaičius, kurio laukia procesai.

**WaitingReturns** – bendros paskirties adresas kur bus įrašomi laukiamo proceso rezultatai.

**Elements** – resurso elementai.

**Parent** – procesas, kuris sukūrė ši resursa.

... - kiti, resurso struktūra priklauso nuo resurso paskirties, vėliau bus aptarta galimos resursų rūšys.

### Elementai

Resursas tai tik deskriptorius aprašantis kažkokį esamo resursą, kurio reiks procesoriams. Tačiau procesoriai iš tiesų kovoja dėl resurso elementų, kurių skaičius gali būti fiksuotas arba ne. Dažnai elementas gali atstoti žinute arba šiaip kažkokį fizini objektą.

Štai čia bendra struktūra:

**Resource** – resursas kuriam priklauso elementas.

**Sender** – procesas, kuris atlaisvino ši elementą.

**Receiver** – procesas, kuris gavo ši atlaisvinta elementą.

... – kiti, elemento struktūra priklauso nuo resurso paskirties, vėliau bus aptarta galimos resursų rūšys.

### Primityvai

Resursas turi keturis primityvus:

* 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.
* Naikinti resursą. Resurso deskriptorius 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 deskriptorius.
* 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.
* 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.

### Paskirstytojas

Resursų paskirstytojo turi keletą tikslų:

* Saugoti visus resursus.
* Aptarnauti procesus kai jie prašo resursų ir nesant jų įtraukti į laukiančių sąrašą bei perleisti procesų stabdymą procesų planuotojui.

## Resursų rūšys

|  |  |  |
| --- | --- | --- |
| Pavadinimas | Procesas kuris kuria | Papildoma struktūra |
| **MOSEnd** | **StartStop** |  |
| **WaitInputStream** | **ReadFromInterface** | ? |

## Procesų rūšys

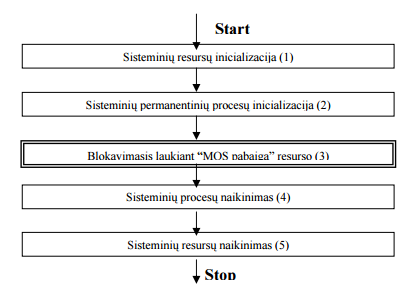
Mūsų modelyje užduoties keliui palaikyti bus naudojami šie procesai:

* **StartStop** – šakninis procesas, sukuriantis bei naikinantis sisteminius procesus ir resursus.
* **ReadFromInterface** – užduoties nuskaitymo iš įvedimo srauto procesas
* **JCL** – užduoties programos, jos antraštės išskyrimas iš užduoties, ir jų organizavimas kaip resursus..
* **JobToSwap** – užduoties patalpinimas išorinėje atmintyje
* **MainProc** – Procesas valdantis **JobGorvernor** procesus.
* **JobGorvernor** – virtualios mašinos proceso tėvas, tvarkantis virtualios mašinos proceso darbą
* **Loader** – iš išorinės atminties duomenys perkeliami į vartotojo atmintį
* **Virtual Machine** – procesas atsakantis už vartotojiškos programos vykdymą.
* **Interrupt** – procesas, apdorojantis virtualios mašinos pertraukimą sukėlusią situaciją.
* **PrintLine** – į išvedimo įrenginį pasiunčiama eilutė iš supervizorinės atminties.

Beveik visi procesai yra sukuriami sistemos darbo pradžioje proceso StartStop. StartStop nekuria tik 2 procesų – JobGorvernor (kiekvienai naujai vartotojo užduočiai MainProc kuria po naują procesą JobGorvernor) ir VirtualMachine, kurį kuria JobGorvernor.

### StartStop

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



### ReadFromInterface

### JCL

### JobToSwap

### MainProc

### Loader

### JobGrovernor

### VirtualMachine

### Interrupt

### PrintLine

Procesą PrintLine kuria ir naikina procesas StartStop. Šio proceso paskirtis – į išvedimo srautą pasiųsti kokioje nors atmintyje esantį pranešimą.

Proceso darbas prasideda blokavimusi dėl “Eilutė atmintyje” resurso(1).Šis resursas turi parametrą, nusakantį iš kurios atminties reikės pasiųsti eilutę į išvedimo srautą, bei atminties adresą, žymintį bloko numerį. Toliau procesas blokuojasi laukdamas leidimo dirbti su kanalų įrenginiu (2), atblokuotas jis nustato kanalo duomenų perdavimo kryptį ir adresą bei įvykdo komandą XCHG (3). Įvykdęs komandą, procesas atlaisvina pranešimą leidžiantį dirbti su kanalu(4).

