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

Turinys

[1 Operacinės sistemos modelis 3](#_Toc483169121)

[1.1 Procesai 3](#_Toc483169122)

[1.1.1 Struktūra 3](#_Toc483169123)

[1.1.2 Primityvai 3](#_Toc483169124)

[1.1.3 Būsenos 3](#_Toc483169125)

[1.1.4 Planuotojas 4](#_Toc483169126)

[1.1.5 Prioritetas 4](#_Toc483169127)

[1.2 Resursai 4](#_Toc483169128)

[1.2.1 Struktūra 5](#_Toc483169129)

[1.2.2 Elementai 5](#_Toc483169130)

[1.2.3 Primityvai 5](#_Toc483169131)

[1.2.4 Paskirstytojas 5](#_Toc483169132)

[1.3 Resursų rūšys 6](#_Toc483169133)

[1.4 Procesų rūšys 6](#_Toc483169134)

[1.4.1 processStartStop 7](#_Toc483169135)

[1.4.2 processExternalMemory 7](#_Toc483169136)

[1.4.3 processManager 7](#_Toc483169137)

[1.4.4 processProgramManager 7](#_Toc483169138)

[1.4.5 processIdle 8](#_Toc483169139)

[1.4.6 processInput 8](#_Toc483169140)

[1.4.7 processOutput 8](#_Toc483169141)

# 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** | **Procesas kuris naudoja** |
| resourceOSStopRequest | processStartStop | processUser |
| resourceMemory | processStartStop | processUser |
| resourceExternalMemoryRequest | processUser | processExternalMemory |
| resourceExternalMemoryRespond | processExternalMemory | processUser |
| resourceProgramManagerRequest | processUser | processProgramManager |
| resourceProgramManagerRespond | processProgramManager | processUser |
| resourceProcessManagerRequest | processUser | processManager |
| resourceProcessManagerRespond | processManager | processUser |
| resourceOutputRequest | processUser | processOutput |
| resourceOutputRespond | processOutput | processUser |
| resourceInputRequest | processUser | processInput |
| resourceInputRespond | processInput | processUser |
|  |  |  |

## Procesų rūšys

Mūsų modelyje užduoties keliui palaikyti bus naudojami šie procesai:

|  |  |  |
| --- | --- | --- |
| **Pavadinimas** | **Procesas kuris sukuria** | **Tikslas** |
| processStartStop | - | Šakninis procesas, sukuriantis bei naikinantis sisteminius procesus ir resursus. |
| processExternalMemory | processStartStop | Atsakingas už failų kontrolė, jų atidarymą, uždarymą, skaitymą, rašymą. |
| processManager | processStartStop | Atsakingas už **processUser** valdymą, kas įeina jų kūrimas bei naikinimas. |
| processProgramManager | processStartStop | Atsakingas už programinio kodo sukūrimą bei kompiliacija. |
| processIdle | processStartStop | Šis procesas leidžiamas tik tada, kai nėra kitų procesu, simuliuoja proceso miegojimą. |
| processInput | processStartStop | Nuskaito tekstą iš konsolės. |
| processOutput | processStartStop | Išveda tekstą iš atminties į konsole |
| processUser | processManager | Vartojo procesas, atlieka bendros paskirties užduotys. |

Beveik visi procesai yra sukuriami sistemos darbo pradžioje proceso processStartStop.

### processStartStop

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

|  |
| --- |
| Laukia **resourceOSStopRequest** |
| Sunaikina **processStartStop** |
| Išjungia realia mašina |

### processExternalMemory

Šis procesas atsakingas už failų valdymą.

|  |
| --- |
| Laukia **resourceExternalMemoryRequest** |
| Atlieka užduotį pagal prašymą: OpenFile/CloseFile/ReadFile/WriteFile |
| Siunčia **resourceExternalMemoryRespond** |
| Kartojama |

### processManager

Šis procesas bene svarbiausias nes jis kuria/naikina vartotojo procesus.

|  |  |
| --- | --- |
| Laukia **resourceProcessManagerRequest** | |
| CreateUserProcess | DestroyUserProcess |
| Siunčia **resourceProgramManagerRequest** | Naikinamas procesas |
| Laukia **resourceProgramManagerRespond** |  |
| Gauna **program** |  |
| Laukia **resourceMemory** |  |
| Sukuria **processUser** su gautu **program** ir **resourceMemory** |  |
| Siunčia **resourceProcessManagerRespond** | |
| Kartojama | |

### processProgramManager

Šis procesas, kuria/naikina **program** iš failų, bei gali sukompiliuoti išeities kodą į **program.**

|  |  |  |
| --- | --- | --- |
| Laukia **resourceProgramManagerRequest** | | |
| SaveProgram | LoadProgram | CompileProgram |
| Siunčia **resourceExternalMemoryRequest** | | |
| Laukia **resourceExternalMemoryRespond** | | |
| Surealizuoja **program** atminti | Surealizuoja **program** iš atminties | Sukompiliuoja perskaityta failą į sukuria **program** iš jo |
| Siunčia **resourceProgramManagerRespond** | | |
| Kartojama | | |

### processIdle

Procesas, kuris simuliuoja procesoriaus neveikimą. Mano projekte jis dar paleidžia **processUser**, kuris yra atmintyje **Startup.exe**.

### processInput

Nuskaito tekstą į atmintį.

|  |
| --- |
| Laukia **resourceInputRequest** |
| Įrašo tekstą. |
| Siunčia **resourceInputRespond** |
| Kartojama |

### processOutput

Išveda tekstą iš atminties į ekraną.

|  |
| --- |
| Laukia **resourceOutputRequest** |
| Išveda tekstą. |
| Siunčia **resourceInputRespond** |
| Kartojama |