|  |
| --- |
| VU MIF 3 KURSAS, 3 GRUPĖ |
| Multi-OS Projektas |
| Procesai ir resursai |
|  |
| **Arūnas Bendoraitis** |
| **05/01/2011** |

|  |
| --- |
|  |



Užduoties sąlyga (G - projektuojama interaktyvi OS):

* Virtualios mašinos procesoriaus komandos operuoja su duomenimis, esančiais steko viršūnėje.
* Yra komandos duomenų persiuntimui iš atminties į steką ir atvirkščiai, aritmetinės (sudėties, atimties, daugybos, dalybos), sąlyginio ir besąlyginio valdymo perdavimo, įvedimo, išvedimo, darbo su bendra atminties sritimi (prieinama visoms vartotojo programoms; komandos leidžia į ją rašyti ir skaityti; sritis apsaugoma semaforais) ir programos pabaigos komandos.
* Registrai yra du: komandų skaitiklio ir steko viršūnės.
* Atminties dydis yra 16 blokų po 16 žodžių (žodžio ilgį pasirinkite patys).
* Realios mašinos procesorius gali dirbti dviem režimais: vartotojo ir supervizoriaus.
* Virtualios mašinos atmintis atvaizduojama į vartotojo atmintį naudojant puslapių transliaciją.
* Yra taimeris, kas tam tikrą laiko intervalą generuojantis pertraukimus.
* Įvedimui naudojama klaviatūra, išvedimui - ekranas.
* Yra išorinės atminties įrenginys - kietasis diskas.
* Vartotojas, dirbantis su sistema, programas paleidžia interaktyviai, surinkdamas atitinkamą komandą.
* Laikoma, kad vartotojo programos yra realios mašinos kietajame diske, į kurį jos patalpinamos „išorinėmis“, modelio, o ne projektuojamos OS, priemonėmis.

# Įžanga

Kaip žinia kompiuteryje gali veikti kelios programos vienu metu. Multiprograminėje operacinėje sistemoje tai pasiekiama procesų perjungimų sistema. Kiekvienas procesas yra vykdomas labai trumpą laiko tarpą ir taip vienoje sekundėje gali būti aptarnauti keli ar keliolika procesų. Tokiu būdu vartotojui atrodo, kad jie tarsi veikia vienu metu. O toliau, viskas šiek tiek detaliau.

# Procesai

## Kas jie tokie?

**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.

## 2.2. 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.

## 2.3. 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 9 perėjimai:

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 gali tapti sustabdytu blokuotu, jei einamasis procesas jį sustabdo, kai jis jau ir taip jau 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 iš blokuoto sustabdyto, jei procesui yra suteikiamas jam reikalingas resursas.

## 2.4. 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 seka galima būtų pavaizduoti tokia diagrama:



Vienas iš planuotojo algoritmų yra prioritetais pagrįstas modelis.

## 2.5. Prioritetai

Kiekvienam procesui suteikiamas prioritetas iš sutarto intervalo. Procesas, turintis didesnį prioritetą vykdomas pirmiau, nei mažesnį prioritetą turintis procesas.

## 2.6. Primityvai

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

# Resursai

## Kas jie tokie?

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.

Resursus galima skirstyti į tokias dvi grupes:

* **Statiniai (daukartinio naudojimo) resursai.** Kuriami sistemos kūrimo metu. Tai mašinos resursai, tokie kaip procesorius, atmintis ar kiti resursai, kurie sistemos veikimo metu nėra naikinami. Jie dar skirstomi į
  + *Skaidomi*, kaip bendroji atmintis, kurio atskiras dalis gali naudoti keli procesai.
  + *Neskaidomi* (procesorius, vaizduoklis, klaviatūra, t.t.) kurie gali būti ‚užimti arba laisvi‘.Procesorius yra ypatingas resursas todėl, kad jis yra reikalingas visiems procesams ir be jo nė vienas procesas netaps vykdomu. Tik vykdomas procesas gali atlikti savo darbą.
* **Dinaminiai (vienkartinio naudojimo) resursai**. Kuriami ir naikinami sistemos darbo metu. Šie resursai naudojami kaip pranešimai. Kartu su jais gali ateiti naudinga informacija.

## Paskirstytojai

Kaip kad procesorius yra skirstomas planuotojo, kiekvienas resursas taipogi yra skirstomas tarn tikro paskirstytojo. Prašydamas resurso ar norėdamas jį atlaisvinti, procesas kreipiasi į atitinkama resursų paskirstytoją kuris privalo jį aptarnauti.

Dėl daukartinio naudojimo resursų konkuruoja procesai. Juos skirstysime eilės principu, tai yra – su kiekvienu resursu susiesime jo laukiančių procesų sąrašą. Tas kuris ankščiau paprašo resurso, ankščiau į tą sąrašą patenka ir gauna tą resursą.  
Taigi pirmiausia resursą gauna pirmasis eilės narys, vėliau jis iš jos išbraukiamas. Nauji procesai, kurie prašo resurso talpinami į eilės galą.

Resursų paskirstytojo paskirtis - suteikti paprašytą resurso elementų kiekį procesui. Resursų paskirstytojo algoritmas gali būti sudėtingas. Pavyzdžiui, turi būti numatyta galimybė atiduoti resurso elementą konkrečiam procesui, arba prašyti kelių resurso elementų. Resurso paskirstytojas peržvelgia visus laukiančius šio resurso procesų sarašą, ir, sutikęs galimybę aptarnauti process perduoda jam reikalingus resurso elementus ir pažymi jį pasiruošusiu. Paskirstytojo pabaigoje yra iškviečiamas planuotojas.

## 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 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.
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.

# 4. Procesų paketas

## 4.1. Apžvalga

Naudosime šiuos procesus:

* **StartStop** – šakninis procesas, sukuriantis bei naikinantis sisteminius procesus ir resursus.
* **ReadFromInterface** – užduoties nuskaitymo iš įvedimo srauto procesas.
* **JobToDisk** – užduotis perkelia vartotojo programą iš supervizorinės atminties į išorinį diską.
* **Checker** – užduoties programos suskaidymas blokais ir jų organizavimas kaip resursus.
* **Loader** – iš išorinės atminties duomenys perkeliami į vartotojo atmintį.
* **MainProc** – procesas, valdantis JobHelper procesus.
* **InputOutput** – procesas darbui su išvedimu bei įvedimu.
* **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.
* **PrintLine** – į išvedimo įrenginį persiunčiama eilutė(spausdintuvas) 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.  
  
Bendrą vaizdą galėtume pavaizduoti tokia schema:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  


## 4.2. StartStop

**StartStop** – šakninis procesas, įjugus kompiuterį pasileidžia automatiškai. Kuria kitus sisteminius procesus bei resursus ir blokuojasi laukdamas pranešimo apie OS darbo pabaigą.



## ReadFromInterface

**ReadFromInterface** – procesas, kopijuojantis duomenų blokus iš išorinės atminties į supervizorinę ir atiduoda tolesniam apdorojimui, kurį atlieka . Jį kuria ir naikina StartStop. Procesas laukia įvedimo srauto iš vartotojo, mūsų atveju tai bus failo pavadinimas kuriame yra programa.



## Checker

**Checker** – procesas, tikrinantis nuskaityto failo struktūros teisingumą





## JobToDisk

**JobToDisk** – sisteminis procesas, perkeliantis duomenis iš supervizorinės atminties į išorinę.





## MainProc

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



Ar vykdymo laikas ne 0?

Taip

## JobHelper

**JobHelper** – valdo vieną procesą „*ViltualMachine*“.



## Loader

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



## VirtualMachine

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



## Interrupt

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

## 

## 4.11 InputOutput

**InputOutput** – procesas darbui su išvedimu bei įvedimu.

