**Operacinės sistemos,**

**reali ir virtuali mašinos**

Turinys

* Virtuali Mašina 2
  + Virtualaus procesoriaus registrai 2
  + Virtualios mašinos atmintis 2
* Reali Mašina 3
  + Realios mašinos registrai 3
  + Realios mašinos atmintys 4
* Komandų sistema 4
* Procesoriaus darbo algoritmas 6
* Pertraukimai 7
* Puslapiavimo mechanizmas 8
* Kanalų įrenginys 9
* Taimerio mechanizmas 9
* Įvedimo, išvedimo įrenginiai 9
* Branduolys 11
* Procesoriaus deskriptorius 11
* Procesai 11
* Procesų būsenos 11
* Procesų deskriptorius 12
* Procesų primityvai 13
* Procesų planuotojas 14
* Resursų deskriptorius 15
* Resursų primityvai 15
* Resursų paskirstytojas 17
* Resursų lentelė 18
* Procesų paketas 19
  + START/STOP 20
  + INTERFACE 20
  + VIRTUAL MACHINE 21
  + SYSTEM IDLE 21
  + INPUT PROGRAM 22
  + INPUT DATA 22
  + OUTPUT DATA 23
  + ADDITIONAL MEMORY 23
  + MAIN PROC 24
  + LOADER 24
  + JOB GOVERNOR 25
  + OUTPUT TO USER 26
  + PERTRAUKIMŲ APDOROTOJAS 27

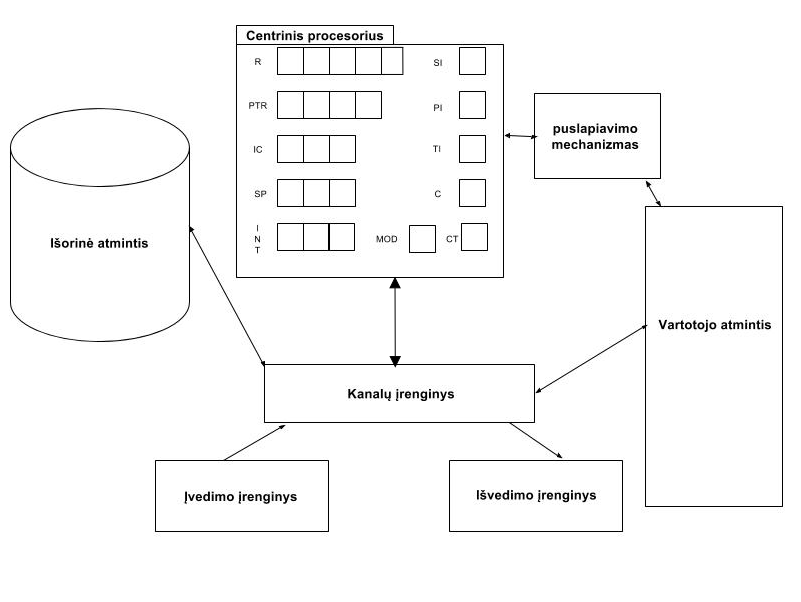
**Virtuali Mašina**



1. **Virtualaus procesoriaus registrai**
   1. R – 5 baitų bendro naudojimo registras
   2. C – 1 baito požymio registras
   3. IC – 3 baitų komandų skaitliuko registras
   4. CT – 1 baito ciklo skaitliuko registras
   5. SP – 3 baitų steko viršūnės registras
2. **Virtualios mašinos atmintis**

Kiekvienai virtualiai mašinai yra skiriama 10 vartotojo atminties blokų. Tuose dešimtyje blokų (100 žodžių) turi tilpti užduoties programa. Kiekvienas virtualios atminties blokas turi virtualų ir realų adresą.

**Reali Mašina**

****

1. **Realios mašinos registrai**
   * MOD - 1 baito procesoriaus būseną nusakantys registras
   * R - 4 baitų bendro naudojimo registras
   * PTR - 4 baitų puslapių lentelės registras
   * IC - 3 baitų komandų skaitliuko registras
   * SP - 3 baitų steko viršūnės registras
   * C - palyginimo požymių registras
   * INT - 3 baitų pertraukimo procedūros adreso registras
   * CT – 3 baito ciklo skaitliuko registras
   * PI - 3 baito programinių pertraukimų registras
   * SI - 3 baito supervisorinių pertraukimų registras
   * TI - 3 baito taimerio mechanizmo registras
2. **Realios mašinos atmintys**

Atmintis – įrenginys informacijai saugoti. Mūsų reali mašina turi dviejų rūšių atmintis: vartotojo ir išorinę.

**Vartotojo atmintis**: skirta virtualių mašinų atmintims, puslapių ir pertraukimų vektoriaus lentelėms laikyti. Mes apibrėšime vartotojo atmintį taip: lentelės dydis – 1000 žodžių po 5 baitus (1 baitas – 8 bitai). 10 žodžių laikysime bloku (takeliu). Taigi vartotojo atmintis lygi 100 blokų, sunumeruotų nuo 0 iki 99, arba 1000 žodžių, sunumeruotų nuo 0 iki 999.

**Išorinė atmintis**: šioje atmintyje saugomi programų tekstai.

Procesorius darbą su atmintimis valdo naudodamas kanalų įrenginį.

Šios atminties dydis yra pastovus – 400 blokų arba 4 000 žodžių (sunumeruotų nuo 0 - 3999) po 5 baitus (1 baitas – 8 bitai).re

6**Komandų sistema**

**Komandų ilgis – 1 žodis (išskyrus komandą CHNGR, kurios komandos ilgis 2 žodziai) .**

Komandos kurias gali vykdyti tiek VM tiek RM

* CHNGR - Žodis esantis adresu [IC] yra komandos kodas, žodis esantis adresu [IC + 1] yra reikšmė kuri priskiriama registrui R. IC = IC + 1
* LRxyz – atminties žodžio, kurio adresas x\*100 + y \* 10 + z turinio kopijavimas į registrą R. R:= [x\*100 + y \* 10 + z]. x, y, z = 0..9.
* SRxyz – registro R reikšmė įrašoma į atminties žodžio, kurio adresas x\*10 + y. [x\*100 + y\*10 + z]:= R, x, y, z = 0..9.
* LOreg - Į registra R įrašoma registre reg buvusi reikšmė. R = reg,  
  reg = {MOD, PTR, IC, SP, C, INT, PI, SI, TI, CT}. Dirbant vartotojo rėžime reg = {IC, C, R, SP, CT}
* ADxyz – prie registro R reikšmės pridedama atminties ląstelės, kurios adresas x\*10 + y, reikšmė. R:= R + [x\*100 + y\*10 + z ]. x, y, z = 0..9.
* SBxyz – iš registro R reikšmės atimamas žodis, kurio adresas x\*100 + y\*10 + z, reikšmė. R:= R - [x\*100 + y\*10 + z]. x, y, z = 0..9.
* MPxyz – registro R reikšmė yra sudauginama su atminties ląstelės, kurios adresas x\*10+y, reikšme. rezultatas padedamas į registrą R. R = [x \* 100 + y \* 10 + z] \* R. x, y, z = 0..9.
* DIxyz - registro R reikšmė dalinama iš atmintimties žodždio [x\*100 + y \* 10 + z]. Į registrą R įrašoma sveikoji dalis.
* CRxyz - jei R = [x\*100 + y\*10 + z], tada C = „T“, kitu atveju C = „F“
* RLxyz - jei R < [x\*100 + y\*10 + z], tada C = „T“, kitu atveju C = „F“
* RGxyz – jei R > [x\*100 + y\*10 + z], tai registrui C priskiriama reikšme „T”, kitu atveju C := „F“
* CZreg – jei reg = 0, tada C = „T“, kitu atveju C = „F“. Reg = { MOD, PTR, IC, SP, C, INT, PI, SI, TI, CT }.
* JCxyz - jei registro C reikšmė = T tai valdymas perduodamas x\*100 + y\*10 + z adresu. IC = x\*100 + y\*10 + z. x, y, z = 0..9
* JPxyz – besąlyginis komandos perdavimas adresu [x\*100 + y\*10 + z];
* CAxyz – komanda call, išaugoma mašinos būsena (į steką išsaugomos reikšmės šia tvarka MOD, PTR C, R, IC, SP, CT) ir IC priskiriama xyz. IC =x\*100 + y\*10 + z. x, y, z = 0..9, jeigu dirbame vartotojo rėžime į steka saugome šiuos registrus C, R, IC, SP, CT.
* PUreg - į [SP] iššaugomas nurodytas registras, steko viršūnė sumažinama vienetu. Rašant reikšmę, ji praplečiama nuliais priekyje jeigu to reikia, jeigu ši komanda vykdoma vartotojo režimu galimi registrai yra: {C, IC, R, SP, CT}.
* POreg – iš [SP] reikšmė įrašoma į nurodytą registrą, steko viršūnė padidinama vienetu. Jeigu reg ={ TI, SI, PI, INT, C, SP, IC, MOD, CT}, tokiu atveju iš pradžios yra nuvaloma tiek baitu kiek reikia suvienodinti atminties ir registro dydi, jeigu ši komanda vykdoma vartotojo režimu galimi registrai yra: {C, IC, R, CT, SP}..
* RETRN – Grįžimo iš proceduros komanda, iš steko grąžinamos registrų reikšmes šia tvarka CT, SP, IC, R, C, PTR, MOD, jeigu dirbame vartotjo režime gražinamos šios reikšmės CT, SP, IC, R, C.
* SYxyz – Programa praneša operacinei sistemai apie veiksmus kurių negali atlikti pati. Supervisorinių pertraukimu registras nurodyta reikšme SI = x \* 100 + y + \*10 + z. x, y, z = 0..9.
* LPxyz – ciklo komanda, sumažina ciklo registro reikšmę vienetu, jeigu CT != 0, vykdomas šuolis į ciklo pradžią JPxyz. x, y, z = 0..9.

Komandos kurias gali vykdyti tik reali mašina

* CHNGM- keičiama mašinos būsena, IF MOD = 1 THEN MOD = 0 ELSE MOD = 1.
* PIxyz – nustatoma registro PI reikšmė nurodytu skaičiumi. PI = x,  
  x = 0..9.
* TIxyz – nustatoma registro TI reikšmė nurodytu skaičiumi. TI = x,  
  x = 0..9.
* PTRxy – nustatoma registro PTR reikšmė, kuri rodys puslapių lentelės bloko numerį. PTR = x\*10+y.  
  x, y = 0..9.
* SPxyz – nustatomas steko viršunės adresas. SP = x\*100+y\*10+z,  
  x, y, z = 0..9.
* INxyz – nustatomas pertraukimų registro adresas. INT = x \* 100+y \* 10 + z. x, y, z = 0..9.
* START - pradedama vykdyti virtuali mašina: išsaugoma realios mašinos busena, virtualios mašinos komandu skaitliukas nunulinamas IC = 0.
* CALLI - kviečiamas pertraukimas, į steką išsaugomos registrų reikšmės šia tvarka MOD, PTR C, R, IC, SP, CT, IC = INT.
* IRETN - vykdomas grįžimas po pertraukimo, grąžinamos buvusios registrų reikšmės iš steko šia tvarka: CT, SP, IC, R, C, PTR, MOD
* BSxyz - nustatomas takelio numeris iš kurio kopijuosime. Kanalų įrenginio registras SB = x \* 100 + y \*10 + z. x, y, z = 0..9
* DBxyz - nustatomas takelio numeris į kurį kopijuosime numeris. Kanalų įrenginio registras DB = x \* 100 + y \*10 + z. x, y, z = 0..9
* STxyz – nustatomas objekto numeris, iš kurio kopijuosime. Kanalų įrenginio registras ST = x \* 100 + y \* 10 + z. x, z, y = 0..9
* DTxyz - nustatomas objekto numeris į kurį kopijuosime. Kanalų įrenginio registras DT = x \* 100 + y \* 10 + z. x, z, y = 0..9
* SZxyz – nurodoma, kiek žodžių bus kopijuojama iš/į objektą(-o). SZ = x \* 100 + y \* 10 + z. x, y, z = 0..9
* XCHGN – vykdomas apsikeitimas duomenimis pagal nustatytus kanalų įrenginio registrus.

**Procesoriaus darbo algoritmas**

1. Procesorius gali dirbti dviem rėžimais: Vartotojo ir supervisoriniame. ( MOD = 1 reiškia, jog dirbama vartotojo, o MOD = 0 – supervisoriaus režime ). Priklausomai nuo rėžimo kuriame dirbame keičiasi aibė komandų kurias gali vykdyti procesorius.
2. Procesorius pasiruošęs vykdyti komandą, nurodytą IC registre. Nuskaitęs komandą, procesorius padidina IC registro reikšmę vienetu. Taip pat, TI reikšmė mažinama vienetu.
3. Vykdoma instrukcija, galimos kelios vykdymo baigtys :
   1. Komanda atpažįstama komandų aibėje – komanda įvykdoma, jeigu reikia nuskaitomi argumentai.
   2. Komanda praneša apie supervisorini pertraukima.
   3. Komanda neatpažįstama arba negalima vykdyti, nustatomas programinio pertraukimo registras.
4. Tikrinami pertraukimų registrai. IF ((SI + PI) = 0 & TI != 0) THEN GOTO1 ELSE pertraukimų apdorotojas (plačiau prie pertraukimų) GOTO1.

**Pertraukimai**

Programinių pertraukimų registras yra PI, supervizorinių registras – SI, taimerio – TI.

Supervisoriniai pertraukimai, vartotojo rėžime, sukuriami SYxyz komandos pagalba,

* Supervisoriniai pertraukimai
  + SI = 1 - virtualios mašinos prašymas gauti duomenis iš įvedimo įrenginio, registre R esanti reikšmė traktuojama: pirmi du baitai nurodo, kiek žodžių bus įrašoma, sekantys du baitai nurodo adresą virtualios mašinos atmintyje.
  + SI = 2 - virtualios mašinos prašymas siųsti duomenis į išvedimo įrenginį. Registre R esanti reikšmė traktuojama: pirmi du baitai nurodo, kiek žodžių bus įšvedama, sekantys du baitai nurodo adresą virtualios mašinos atmintyje.
  + SI = 3 - virtualios mašinos prašymas išskirti papildomos atminties, tiek blokų kokia reikšmė yra registre R, ne daugiau kaip 10 bloku.
  + SI = 4 - virtualios mašinos pranešimas apie darbo pabaigą.
  + SI = 5 - vartotojo nurodymas išjungti sistemą.
  + SI = 6 – vartotojo prašymas parašyti užduotį
* Programiniai pertraukimai:
  + PI = 1 – atminties apsaugos pažeidimas.
  + PI = 2 – neleistinas operacijos kodas.
  + PI = 3 – atminties trūkumas
  + PI = 4 – neleistinas priskyrimas
* Esant TI = 0, bus iššaukiamas taimerio pertraukimas.

Aptikus pertraukimą išsaugoma esamos mašinos būsena steke (MOD, PTR C, R, IC, SP, CT), jeigu reikia pakeičiamas režimas, kviečiamas pertraukimų apdorotojas.

**Puslapiavimo mechanizmas**

Realios mašinos vartotojo atmintis siekia 100 takelių (arba blokų). Kiekvienai naujai sukurtai virtualiai mašinai maksimaliai galima skirti 10 takelių iš 100. Jie gali būti parinkti bet kokiu būdu. Klausimas: kaip virtuali mašina gali sužinoti kokio nors jai priklausančio takelio realų adresą? Tam naudosime puslapiavimo mechanizmą.

Puslapiavimo mechanizmo esmė: sakykime, kuriama nauja virtuali mašina. Jai reikia dešimt takelių atminties. Mes parinkome takelius su numeriais: 1, 3, 5, 7, 9,11, 13, 15, 17, 19. Šiais takeliais naudosis virtuali mašina. Pati virtuali mašina šiuos takelius mato sunumeruotus nuo 0 iki 9. t.y. 1 takelis jai yra nulinis, 5 takelis jai yra antras, o 19 takelis – devintas. Kaip išlaikyti sąryšius tarp realių ir virtualių takelių adresų? Naudosime puslapių lentelę. Puslapių lentelė – tai vienas takelis (t.y. 10 žodžių). Kiekvieno žodžio eilės numeris atitiks virtualios mašinos takelio numerį, ir jame (žodyje) bus laikomas realus to takelio numeris.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Realus takelio adresas | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 |
| Virtualaus takelio adresas | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Taigi, dabar virtuali mašina, norėdama sužinoti realų takelio adresą, kreipiasi į savo puslapių lentelę ir nuskaito reikšmę, esančią žodyje su takelio numeriu. Pvz. penkto virtualios mašinos takelio realus numeris yra 11. Dabar turi išaiškėti procesoriaus registro PTR prasmė. Juk puslapių lentelė taip pat yra atminties takelis. Ir šis takelis, be abejo, taip pat turi savo realų numerį. Todėl registro PTR reikšmė – puslapių lentelės adresas. Kiekviena virtuali mašina, prieš pradėdama darbą, nustatys šį registrą jai reikalinga reikšme.

Dabar galime apžvelgti patį registrą PTR. Tai 4 baitų registras. Simboliškai pažymėsime PTR reikšmę a0a1a2a3, kur ai yra baitai. Baitas a0 yra nenaudojamas, a1 puslapių lentelės ilgis atėmus 1 (arba nenaudojamas. Tai priklauso nuo puslapiavimo mechanizmo realizacijos), o štai a2 \* 10 + a3 žymi puslapių lentelės adresą. Dabar galime pateikti formulę, kuri virtualiam adresui x1x2 gražina realų adresą:

**Realus adresas = 10\*[10\*(10 \* a2 + a3) + x1] + x2**

**Kanalų įrenginys**

Kanalų įrenginys leidžia dirbti su atmintimis. Priklausomai nuo nustatytų registrų kanalų įrenginys gali vykdyti apsikeitimą duomenimis visomis galimomis kryptimis.Veiksmai su kanalų įrenginiu atliekami tik supervizoriaus režime. Dabar bus pateikta kanalų įrenginio vartotojo sąsaja:

Kanalų įrenginio registrai:

* SB: Takelio, iš kurio kopijuosime numeris.
* DB: Takelio, į kurį kopijuosime numeris
* ST: Objekto, iš kurio kopijuosime, numeris
  + 1. Vartotojo atmintis;
  + 2. Išorinė atmintis;
  + 3. Įvedimo srautas;
* DT: Objekto, į kurį kopijuosime, numeris
  + 1. Vartotojo atmintis;
  + 2. Išorinė atmintis;
  + 3. Išvedimo srautas;
* SZ - žodžių kiekis kiek kopijuosime

Kartu kanalų įrenginys turi komandą XCHG, tačiau neturi procesoriaus, kuris galėtų ją įvykdyti. Šią komandą vykdo centrinis procesorius, taigi, šis kanalų įrenginys nėra lygiagrečiai su centriniu procesoriumi veikianti aparatūra.

Procesas, norėdamas pasinaudoti kanalų įrenginiu, turi nustatyti kanalų įrenginio

registrus ir tada įvykdyti komandą XCHG.

**Taimerio mechanizmas**

Laikysim, kad įvedimo/išvedimo instrukcijos atliekamos per 3 laiko momentus, visos kitos per 1 laiko momentą. Dabar apie veikimo principą.

Pradedant virtualios mašinos užduoties vykdymą TI registro reikšmė nustatoma tam tikrai reikšmei. Tarkim TI = 10. Įvykdžius eilinę instrukciją TI reikšmė mažinama priklausomai nuo to per kiek laiko momentų ši instrukcija yra atliekama.

**Įvedimo, išvedimo įrenginiai**

Įvedimo - klaviatūra, Išvedimo - monitorius.

Branduolys

Operacinės sitemos branduolį sudaro šios komponentės:

* Procesoriaus deskriptorius
* PS - visų procesų sąrašas
* RES – visų resursų sąrašas
* Planuotojas - atsakingas už procesoriaus skirstymą
* Pertraukimų apdorotojas - atsakingas už kilusiu pertraukimų apdorojimą
* Paskirstytojai - atsakingi už resursų skirstymą
* Procesų primityvai
* Resursų primityvai
* Procesų deskriptoriai

**Procesoriaus deskriptorius**

* Count - procesų skaičius
* Proc - vykdomo proceso vardas

Procesai

Procesas tai programa esanti tam tikroje veiklumo stadijoje kurią nusako proceso deskriptorius. Procesai gali būti kuriami ir naikinami sistemos darbo metu. Vieni procesai gali kurti kitus. Kiekvienas procesas kovoja dėl sisteminių resursų, tačiau vieni yra svarbesni už kitus, tam naudojami procesu prioritetai. Procesas negali būt vykdomas be procesoriaus resursų, jį gali gauti tik turėdamas visus kitus jam reikalingus resursus.

**nProcesų būsenos**

Procesas yra vykdymo būsenoje iki kol įvyksta pertraukimas arba einamasis procesas nepaprašo papildomų resursų, arba pats savęs nesustabdo.

Procesu būsenoms keisti naudojami primityvai:

* Stabdyti procesą
* Aktyvuoti procesą
* Praįyti resurso
* Atlaisvinti resursą

Procesas sukuriamas primityvu Kurti Procesą. Proceso buseną gali keisti ir planuotojas. (pavaizduota apačioje)

**Galimos procesų būsenos** :

* Vykdomas – tai procesas, turintis procesorių ir galintis vykdytį savo užduotį;
* Blokuotas – tai procesas, prašantis resurso;
* Pasiruošęs – tai procesas, turintis visus resursus, išskyrus patį procesorių;
* Sustabdytas – tai procesas, sustabdytas dėl kito proceso veiklos.

**Procesu deskriptorius**

* ID[i] - Proceso vardas.
* CPU[i] - Vykdomo proceso, procesoriaus būsena.
* OA[i] - Operatyvios atminties resuras.
* R[i] - Visi kiti proceso turimi resursai.
* SD[i] - Nuoroda į sąraša kuriame yra procesas (LPS, PPS)
* SR[i] - Proceso sukurtų resursų sąrašas.
* ST[i] - Proceso būsena.
* T[i] - Nuoroda į proceso tevą.
* S[i] - Nuorodą i sūnų sąrašą.
* PR[i] - Proceso prioritetas.

**Procesų primityvai**

Kurti Procesą - yra sukuriamas naujas procesas

**KurtiProcesą** ( n, S0, M0,R0,K0) vardas/ procesoriaus busena/ operatyvios atminties resursas / kiti resursai / prioriteteas

{

ID[i] = n (pagal parametra)

CPU[i] = S0 (pagal parametra)

OA[i] = M0 (pagal parametra)

R[i] = R0 (pagal parametra)

SR[i] = NULL (procesas dar nėra sukūręs resursų, todėl sąrašas yra tuščias)

PR[i] = K0 (pagal parametra)

ST[i] = READY (nustatoma būsena: pasiruošęs

SD[i] = PPS (nurodoma kad procesas bus pasiruošusių procesu sąraše)

Įrašyti(PPS, i)

T[i] = THIS (nurodoma, kad proceso tėvas yra šiuo metu vykstantis procesas

Įrašyti (S[THIS] , i) (į šiuo metu vykdomo proceso vaikų saraša itraukiamas naujai sukurtas procesas)

S[i] = NULL (procesas dar neturi savo sukurtų procesu todel sąrašas tuščias)

}

**NaikintiProcesa**(i) - sunaikinamas procesas

{

NUTRAUKTAS\_VYKDOMAS\_PROCESAS = false

nutraukti (i);

If NUTRAUKTAS\_VYKDOMAS\_PROCESAS then PLANUOTOJAS

}

**nutraukti**(i)

{

If ST[i] = RUN then NUTRAUKTAS\_VYKDOMAS\_PROCESAS = true

Pasalinti (SD[i], i)

For ALL s FROM S[i] DO nutraukti(s)

For ALL r FROM (OA[i] || R[i] DO)

IF PNR(r) THEN Įrašyti (PA[r], Pašalinti(OA[i] || R[i]) )

For All r FROM SR[i] DO

NRD(i)

NPD(i)

}

**StabdytiProcesa**(i) - procesas sustabdomas

{

EINAMAS = (ST[i] == RUN)

IF EINAMAS THEN STOP(i) (procesoriaus atėmimas iš proceso, išsaugoma procesoriaus būsena į nutraukiamo proceso deskriptoriu)

IF ST[i] == (BLOCKED || BLOCKEDS) THEN ST[i] = BLOCKEDS

ELSE ST[i] = READYS

IF EINAMAS THEN PLANUOTOJAS (jeigu buvo nutrauktas einamas procesas reikia kviesti planuotoja nes procesorius nepriskirtas jokiam procesui

**AktyvuotiProcesa** - procesas aktyvuojamas

{

IF ST[i] = readys THEN ST[i] = ready else ST[i] = blocked

If ST[i] = ready then PLANUOTOJAS

}

**Procesu planuotojas**

Planuotojas tai proceso resurso paskirstytojas, jis planuoja procesų veiklą pasiremdamas jų prioritetu. Jo veiksmų seka nurodoma diagramoje apčioje.



Resursai

**Resursu deskriptorius**

* RS – išorinis resurso vardas ; ID.
* PNR[i] – nurodo, ar resursas yra pakartotinio panaudojimo, galimos reikšmės { true, false }.
* K[i] – resursą sukūręs procesas.
* PA[i] – prieinamų resursų sąrašas.
* SU[i] - sąrašas suvartotų resursų
* LPS[i] – laukiančių procesų sąrašas.
* INFO[i] - papildoma informacija kuri gali būti reikalinga procesams
* PASK( I, K, L ) – resurso paskirstytojo programos adresas.
  + K – kiek procesų aptarnavo;
  + L – procesų vardai.

**Resursu primityvai**

* **Kurti resursą.**
  + RS – resurso išorinis vardas
  + PN – ar pakartotino naudojimo resursas
  + PA – prieinamumo aprašymas
  + LPS – laukančių procesų sąrašo antraštė
  + PASK – resurso paskirstytojo programos adresas

Sukuria VV ir LPS[VV],

Kurti\_Resursa( RS, PN, PA, LPS, PASK ){

i = RVV # naujas resurso VV

RS[i] = RS

PN[i] = PN # Nurodoma, ar resursas yra pakartotinio panaudojimo

K[i] = THIS # Resurso tėvo VV. Esamasis procesas, kuriame ir panaudojamas šis primityvas

PA[i] = PA #Nurodomas prieinamumo sąrašas

LPS[i] = LPS #Nurodomas laukiančių procesų sąrašas

PASK[i] = PASK #Nurodomas resurso paskirstytojo programos adresas

Įjungti(SR[\*],i); #į šiuo metu vykdomo proceso sukurtų resursų sąrašą įtraukiamas naujas sukurtas resursas

}

* **Naikinti resursą.** Sunaikinti resursą gali jo tėvas arba pirmtakas. Panaikinamas resurso deskriptorius, prieš tai suaktyvinant jo laukiančius procesus.
  + RS – resurso išorinis vardas

Naikinti\_Resursa( RS ){

r = VV(RS)

R = Pašalinti( LPS[r] ) #gaunama nuoroda į pašalinamą elementą

While R <> ø {

R = Pašalinti( LPS[r] )

}

NaikintiRD[RS] #Naikinamas resursų deskriptorius

Planuotojas()

}

* **Prašyti resurso.** Šis primityvas kviečiamas, kai procesui trūksta resurso. Iškvietus primityvą proceso būsena tampa „blokuotas“ ir taip pat jis įtraukiamas į laukiančiųjų sąrašą.
  + RS – resurso išorinis vardas
  + D – kokios resurso dalies prašoma
  + PR – Proceso prioritetas

Prašyti\_resurso( RS, D, PR ){

r = VV(RS)

Įjungti( LPS[r], (THIS ,D, PR) ) #Procesas „įjungiamas“, įtraukiamas į resurso laukiančiųjų sąrašą. THIS – šiuo metu vykdomas procesas.

PASK( r, K, L ) #K – kiek aptarnautų resursų ; L – aptarnautų procesų vidinių vardų sąrašą. L – rezultatas.

B = true #požymis tikrinimui, ar yra dirbame su einamuoju procesu

FOR J = 1 STEP 1 UNTIL K DO

If L[J] <> THIS {

i = L[J]

Įjungti( PPS, i )

SD[i] = PPS

ST[i] = if ST[i] = blocked then READY

else READYS

}

ELSE B = false #reiškia, tai yra einamasis proceas

IF B{

ST[THIS] = BLOCK

SD[THIS] = LPS[i]

PROC[P[i]] = ø

Pašalinti(PPS, THIS)

}

Planuotojas()

}

* **Atlaisvinti resursą**
  + RS – išorinis vardas
  + D – atlaisvinamos resurso dalies aprašymas

AtslaisvintiR( RS, D, I ) RS - resursas, D - resurso dalis, I - išvietęs procesas{

r = VV(RS)

If PNR[r] == true then

Įjungti( PA[r], D) #Resurso prieinamumo aprašymas pakeičiamas

Pašalinti( R[I], r, D) #Pašalinamas iš proceso resursų sąrašo

PASK( r, &K, &L ) #paskirstytojo programa, K – aptarnaujančių procesų skaičius, L – aptarnaujamų procesų vidaus vardų masyvas

FOR J = 1 STEP 1 UNTIL K DO{

IF L[J] <> Ø{

i = L[J]

Įjungti( PPS, i ) #Procesas įtraukiamas į pasiruošusių procesų sąrašą

SD[i] = PPS

ST[i] = if ST[i] = blocked then READY

else READYS

}

}

IF K <> 0 Planuotojas

else if PNR[r] == false{

Įjungti( SU[r], D) #Resursas įtraukiamas į suvartotų resursų sąrašą

Pašalinti( R[I], r, D) #Pašalinamas iš proceso resursų sąrašo

}

}

* **Aktyvuoti resursą**
  + r – resursas
  + D – aktyvuojamo resurso dalies aprašymas
  + MSG – pranešimas

Aktyvuoti (r, D, MSG)

{

pridįti(INFO[r], D, MSG)

For all r FROM INFO

{

Pridėti (PA[r],D, MSG )

pasalinti(INFO[r], D, MSG)

}

}

* **Deaktyvuoti resursą**
  + r – resursas

Deaktyvuoti(r)

{

pasalinti (SU[r], ALL)

}

**Resursų paskirstytojas**

Paskirstytojas pagal duotą resursą, eina į prieinamų resursų sąrašą. jeigu jame yra prašoma reikalinga resurso dalis, ji priskiriama proceso resursams

PASK(r, K, L)

{

For a = 0 to SIZE(LPS) do

{

If PA[r].Resusrso dalis >= LPS[r].resurso dalis

i = LPS[r] procesas

Įrašyti (r[i], r, PA[r.resurso dalis]

Pasalinti (PA[r], resurso dalis)

Pasalinti(LPS[r], i)

L[K] = i

K++

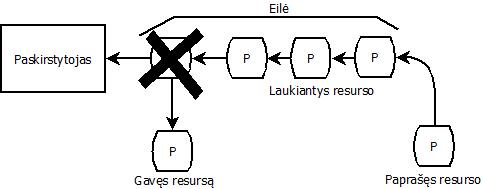
}

}

Resursų paskirstytojas organizuoja procesų darbo palaikymą, teikiant reikiamus resursus.

Procesai konkuruoja dėl *daugkartinio panaudojimo* resursų. Norint išvengti konfliktų, procesams yra suteikiami prioriteto indeksai, kurie parodo jų svarbumą laukiančiųjų eilėje. Kiekvienam resursui yra sudaromas procesų sąrašas, kuriems jis yra reikalingas. Sąrašas yra formuojamas įtraukiant procesus su jų prioritetų indeksais bei išimant procesą, kai procesas atlieka darbą su duotu resursu. Nauji procesai į resursų laukiančiųjų sąrašą įtraukiami pagal prioriteto indeksą.

*Vienkartinio resurso* panaudojimo atveju, stebima, kada atsiranda reikiami resursai, taip pat informuojant procesus, kuriems to reikia. **!!!! PAPILDYTI**



**Resursų lentelė**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resurso pavadinimas** | **Resurso grupė** | **Procesas, kuris sukuria** | **Procesas, kuris sunaikina** | **Procesas, kuris prašo** | **Procesas, kuris atlaisvina** |
| **Įvedimo įrenginys** | Daugkartinio | StartStop | StartStop | Interface, Input program, Input data | Interface,Input program, Input data |
| **Išvedimo įrenginys** | Daugkartinio | StartStop | StartStop | Output\_data,  Output to user | Output\_data,  Output to user |
| **Operatyvioji atmintis** | Daugkartinio | StartStop | StartStop | Additional memory, Job Governor | - |
| **Išorinė atmintis** | Daugkartinio | StartStop | StartStop | Input program | Input program |
| **Išorinis atminties įrenginys** | Daugkartinio | StartStop | StartStop | Input program,  Loader | Input program,  Loader |
| **Laukimas** | Daugkartinio | StartStop | StartStop | System idle | Sysitem idle |
| **Darbo\_pabaiga** | Vienkartinis | Output to user | - | StartStop | - |
| **VM\_Interrupted** | Vienkartinis | Interrupt | Job Governor | Job Governor | Interrupt, Job Governor |
| **Pranešimas\_**  **vartotojui** | Vienkartinis | Interrupt, Job Governor | Job Governor | Output to user | Interrupt, Output to user |
| **Įvesk\_ programą** | Vienkartinis | Output to user | Output to user | Input program | Input program |
| **Įvesk\_programą\_end** | Vienkartinis | Input program, Input data | Input program | Output to user | Output to user |
| **Input\_data** | Vienkartinis | Job Governor | Job Governor | Input data | Input data |
| **Input\_data\_end** | Vienkartinis | Input program, Input data | Input data | Job Governor | Job Governor |
| **Output\_data** | Vienkartinis | Job Governor | Job Governor | Output\_data | Output\_data |
| **Output\_data\_**  **end** | Vienkartinis | Output\_data,  Output to user | Output\_data,  Output to user | Job Governor | Job Governor |
| **Ask\_memory** | Vienkartinis | Job Governor | Job Governor | Additional memory | Additional memory |
| **Memory\_given** | Vienkartinis | Additional memory | Job Governor | Job Governor | Job Governor |
| **Užduotis\_**  **paimta** | Vienkartinis | Job Governor | Job Governor | Main Proc | Main Proc |
| **Užduotis\_**  **Išorinėje\_**  **atmintyje** | Vienkartinis | Job Governor,  Input program | Main Proc | Main Proc | Main Proc |
| **Pakrovimo paketas** | Vienkartinis | Job Governor | Job Governor | Loader | Loader |
| **Loader\_end** | Vienkartinis | Loader | Loader | Job Governor | Job Governor |
| **Output\_to\_user\_end** | Vienkartinis | Output to user | Output to user | Job Governor | Job Governor |

Procesų paketas

**Procesų prioritetai**

|  |  |
| --- | --- |
| **Procesai** | **Prioritetai** |
| START/STOP | 0 |
| OUTPUT\_TO\_USER | 1 |
| INPUT PROGRAM, INPUT DATA, OUTPUT DATA, ADDTIONAL MEMORY, MAIN PROC, LOADER | 2 |
| JOB\_GOVERNOR | 3 |
| INTERFACE, VIRTUAL\_MACHINE | 4 |
| SYSTEM IDLE | 5 |

Sistemą sudaro šie procesai:

* START/STOP - šakninis procesas, sukuriantis sistemai reikalingus procesus ir resursus, darbo pabaigoje butent šis procesas viską sunaikina
* INTERFACE - procesas atsakingas už vartojo įsiterpimą į sistemos darbą
* INPUT PROGRAM -procesas atsakingas už vartotojo programos įvedima iš išorinio įrenginio ir patalpinimą į išorinę atmintį
* INPUT DATA - procesas atsakingas už Virtualios mašinos įvedimo prašymą
* OUTPUT DATA - procesas atsakingas už Virtualios mašinos išvedimo prašymą
* ADDTIONAL MEMORY - procesas atsakingas už Virtualios mašinos atminties prašymą
* MAIN PROC - Procesas atsakingas už JOB\_GOVERNOR proceso sukūrimą
* JOB\_GOVERNOR - Procesas atsakingas už virtualios mašinos proceso sukūrimą ir darbo priežiūrą
* VIRTUAL MACHINE - Procesasatsakingas uz Virtualios mašinos vykdymą
* OUTPUT\_TO\_USER - procesas atsakingas už sisteminių pranešimų vartotojui spausdinimą
* LOADER - procesas atsakingas programos perkėlimą iš išorinės atmienties į operatyviąją
* SYSTEM IDLE - procesas skirtas užimit procesorių jeigu nėra kitų pasiruošusių procesų

**START/STOP INTERFACE**



**VIRTUAL\_MACHINE**



**SYSTEM IDLE**

****

**INPUT PROGRAM**



**INPUT DATA**



**OUTPUT DATA** 

**ADDITIONAL\_MEMORY**

**MAIN PROC**

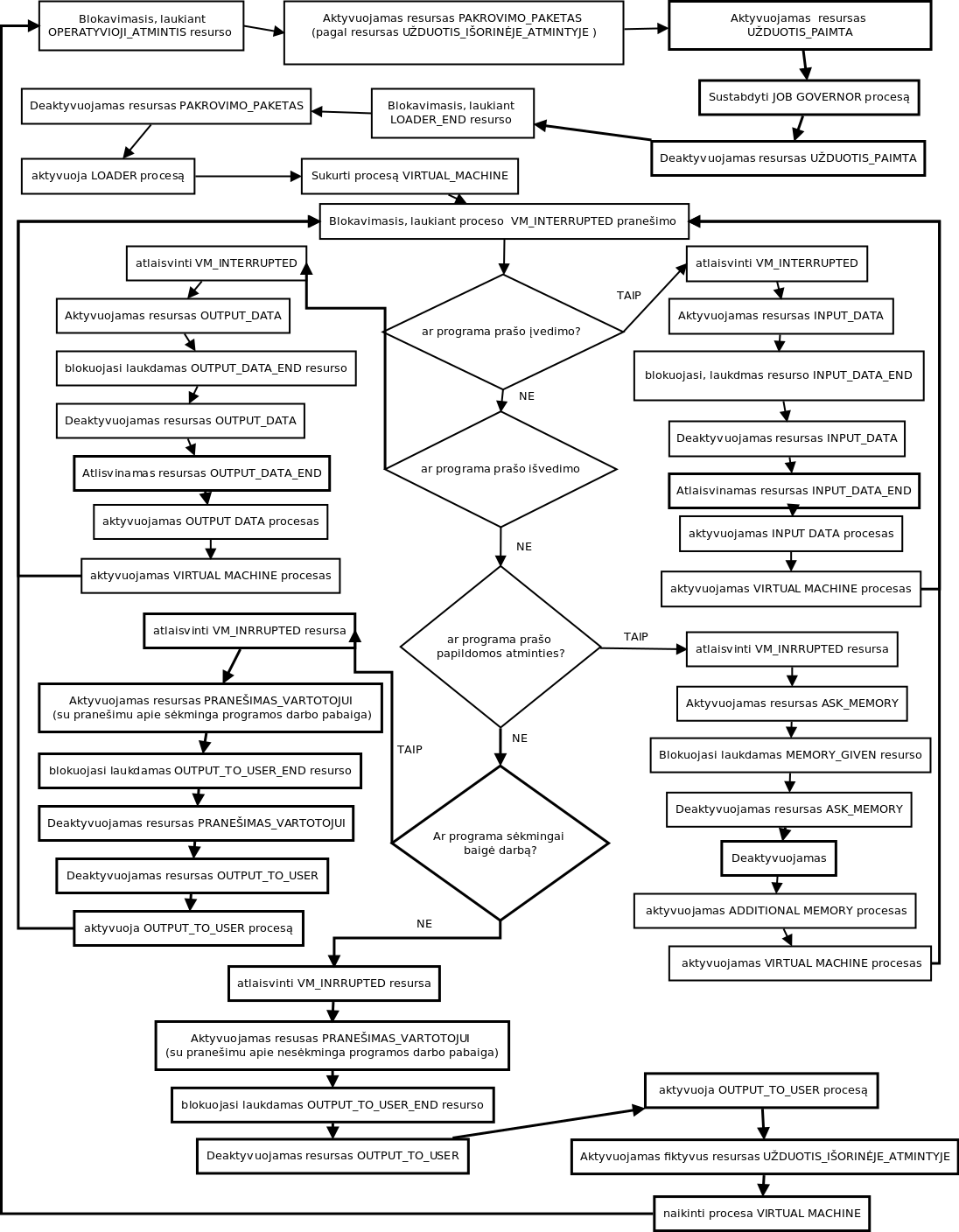


**LOADER**



**OUTPUT TO USER**

****



**Pertraukimų Apdorotojas**

