VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Multiprograminės operacinės sistemos projektas

Parengė: Darius Babravičius, Simonas Jončys, Gytis Karčiauskas

Versija 5.0

2011.05.17

Turinys

[1.1. Procesų valdymas 3](#_Toc292812174)

[1.2. Proceso deskriptorius (PD) 3](#_Toc292812175)

[1.3. Procesų būsenos 4](#_Toc292812178)

[1.4 Procesų paketas 6](#_Toc292812179)

[• PrintLine – į išvedimo įrenginį pasiunčiama eilutė iš supervizorinės atminties. 6](#_Toc292812180)

[1.5. Planuotojas 6](#_Toc292812181)

[1.6. Planuotojo darbas 7](#_Toc292812182)

[1.7. Planuotojo veiksmų seka 7](#_Toc292812183)

[1.8. Sisteminiai ir vartotojiški procesai 8](#_Toc292812184)

[1.9 Procesų primityvai 9](#_Toc292812185)

[2.1 Resursai 18](#_Toc292812186)

[2.2 Resurso deskriptorius 18](#_Toc292812187)

[2.3 Resurso primityvai 19](#_Toc292812188)

**Projektuojama interaktyvi OS (D)**

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 ir programos pabaigos komandos. Registrai yra du: komandų skaitiklio ir steko viršūnės. Atminties dydis yra 256 blokų po 256 žodžius (žodžio ilgį pasirinkite patys).

Realios mašinos procesorius gali dirbti dviem režimais: vartotojo ir supervizoriaus. Virtualios mašinos atmintis atvaizduojama į vartotojo atmintį naudojant puslapių transliaciją. Numatytas swapping mechanizmo palaikymas, t.y. vartotojo atminties puslapiai gali būti iškeliami į išorinę atminti. 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. Trūkstant atminties, sistema dalį vartotojo atminties puslapių (bet ne visą kažkurios VM atmintį) perkelia į išorinę atmintį, t.y. naudoja swapping'ą.

Į OS galima žiūrėti kaip į didelę ir sudėtingą sistemą, sudarytą iš dalių, kurios valdo:

* procesus;
* resursus (CPU ir atmintį);
* failus.

# Procesų valdymas

Programos tekstas yra įvairių instrukcijų seka, o jos vykdymas – dinamiškas procesas. Programa yra pasyvi, kol saugoma išorinėje atmintyje, ir tada ji neturi jokios įtakos kompiuteriui. Programa tampa aktyvi, kai įrašoma į operatyviąją atmintį ir OS ją paleidžia. Aktyvi programos būsena vadinama procesu. Taigi procesas - tai kokioje nors veiklumo stadijoje esanti programa. Kiekvienas procesas gali patekti į įvairias būsenas: jis gali būti vykdomas, blokuotas, atidėtas, paruoštas vykdyti ir t.t. Atskiras proceso būsenas, perėjimą iš vienos į kitą, taip pat nuo vieno proceso prie kito, valdo OS. Pačios OS darbas yra įvairių procesų seka. Paprastai sistemoje vienu metu vyksta daug procesų, kurių dalis gali būti OS procesai (vykdo sistemines užduotis), o kiti – vartotojo procesai (vykdo vartotojo taikomąją programą). Kai kuriems procesams gali prireikti tų pačių kompiuterio resursų, todėl OS turi juos tinkamai paskirstyti.

# Proceso deskriptorius (PD)

PD – proceso veiklumo stadiją apibūdinantis proceso aprašas. Apraše ir yra laikomi visi procesui reikalingi parametrai: virtualus procesorius, registrų reikšmės ir jam reikalingi kintamieji. Procesų aprašai dinaminiai objektai – jie gali būti sukurti/sunaikinti sistemos veikimu metu. Realiai procesą, kaip ir resursą operacinėje sistemoje atstovauja deskriptorius. PD – tai tam tikra struktūra (ne masyvas) – jei kalbame apie visų procesų deskriptorius, tai turime struktūrų masyvą, kur *i* proceso vidinis vardas – nurodytų struktūros numerį masyve. PD – susideda iš komponenčių, kurioms priskiriame vardus:

1. Id[i]– proceso išorinis vardas, reikalingas statiniams ryšiams tarp procesų nurotyti.
2. Mašina – čia turime omeny procesą vykdančio procesoriaus apibūdinimą.
   1. *CPU[i]* – apibūdina centrinio procesoriaus būseną ir teises vykdant procesą. Kai

proceso vykdymas nutraukiamas, proceso būsena išsaugoma;

* 1. *P[i]* – identifikuoja procesorių;
  2. *OA[i]* – operatyvi atmintis turima i-tojo proceso (nuoroda į sąrašą);
  3. *R[i]* – *i-jo* proceso turimi resursai – informacija, kokius resursus yra gavęs procesas. Nuoroda į sąrašą, kuriame yra išvardinta resursai;
  4. *SR[i]* – Sukurtų resursų sąrašas (resursų deskriptorių).

1. informacija apie proceso būseną:
   1. ST[i] – proceso statusas (RUN, READY, READYS, BLOCK, BLOCKS)
   2. SD[i] – nuoroda į sąrašą kuriame yra procesas (PPS – pasiruošusių procesų sąrašas, LPS – laukiančių proc.sar.);
2. proc.sąryšis su kitais procesais:
   1. T[i] – i-tojo proceso tėvas (tėvinio proc.vidinis vardas);
   2. S[i] – nuoroda į i-tojo proceso vaikinių proc.vidinių vardų sąr;
3. PR[i] – prioritetas.

# Procesų 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 vistiek negali tęsti savo darbo be reikiamo resurso). Galime išskirti šias procesų būsenas:

* vykdomas – turi procesorių;
* blokuotas – prašo resurso (išskyrus procesoriaus);
* pasiruošęs – vienintelis trūkstamas resursas yra procesorius;
* sustabdytas – kito proceso sustabdytas procesas (kai procesui negalima leisti gauti procesoriaus, nors jis ir yra pasiruošęs).

Vykdomas

Blokuotas

Blokuotas sustabdytas

Pasiruošęs

Pasiruošęs sustabdytas

Būsenų elgesys:

* vykdomas procesas blokuojamas jam prašant ir negavus resurso;
* vykdomas procesas tampa pasiruošusiu atėmus iš jo procesorių;
* blokuotas procesas tampa pasiruošusiu, kai yra suteikiamas reikalingas resursas;
* pasiruošę procesai varžosi tarpusavyje. Gavęs procesorių, procesas tampa vykdomas;
* procesas gali tapti sustabdytu blokuotu, jei einamasis procesas jį sustabdo, kai jis jau ir taip yra blokuotas;
* procesas tampa blokuotu iš blokuoto sustabdyto, jei einamasis procesas nuima būseną sustabdytas;
* procesas gali tapti pasiruošusiu sustabdytu, jei einamasis procesas jį sustabdo, kai jis yra pasiruošęs;
* procesas tampa pasiruošusiu iš pasiruošusio sustabdyto, jei einamasis procesas nuima būseną sustabdytas;
* procesas tampa pasiruošusiu sustabdytu iš blokuoto sustabdyto, jei procesui yra suteikiamas jam reikalingas resursas.

# Procesų paketas

* **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.

# Planuotojas

Planuotojo paskirtis yra atimti 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, kad kiekvienas procesas būtų vykdomas maksimaliu prioritetu (jei prioritetai vienodi, tai vykdomas tas, kuris sąraše pirmesnis);
* laikyti procesorių užimtą netoli 100 %;
* sumažinti iki minimumo atsakymo laiką vartotojams;
* maksimizuoti darbų skaičių nuveiktą per laiko vienetą.

# Planuotojo darbas

Planuotojo darbas skirstomas Į tris etapus:

1. surasti procesą su aukščiausiu prioritetu;
2. surasti neaktyvų procesorių ir jį išskirti (skirti pasiruošusiamprocesui);
3. jei procesorių nėra, peržiūrėti vykdomus procesus, ir, jei jų prioritetai mažesni, atiduot procesorių pasiruošusiems (perskirstymas).

# Planuotojo veiksmų seka

Planuotojas kviečiamas (iš branduolio primityvų), kai norima pakeisti einamąjį procesą kitu (susiklosčius tinkamoms aplinkybėms einamasis procesas gali ir nepasikeisti, ypač, jei jo prioritetas aukštas).

Ar einamas procesas blokuotas?

Ar yra pasiruošusių procesų?

Procesas įtraukiamas į blokuotų procesų sąrašą

Vykdomas procesas

Imamas pirmas pasiruošęs procesas ir suteikiamas jam valdymas

Laukiama, kol atsiras pasiruošęs procesas

Taip

Taip

Ne

Ne

Pirmas planuotojo žingsnis yra einamojo proceso būsenos tikrinimas. Jei einamasis procesas nėra blokuotas, jis įtraukiamas į pasiruošusių procesų sąrašą. Sekantis planuotojo žingsnis yra tikrinimas, ar yra pasiruošusių procesų. Jeigu pasiruošusių procesų sąrašas nėra tuščias, tai imamas pirmasis sąrašo elementas. Tai bus procesas su aukščiausiu prioritetu. Sekantis žingsnis – valdymo perdavimas. Naujasis procesas pažymimas pažymimas kaip einamasis.

# Sisteminiai ir vartotojiški procesai

Procesus galima suskirstyti į sisteminius ir vartotojiškus. Sisteminis procesas – tai procesas, atliekantis tam tikras sistemines funkcijas, pavyzdžiui, vartotojiško proceso palaikymą. Vartotojiškas procesas skirtas vartotojo užduočiai atlikti. 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 įvykdoma vartotojo užduotis. Vartotojo užduočiai baigus darbą, neužsiblokavę procesai ne už ilgo užsiblokuoja, ir vėl prasideda laukimas.

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. Paprastai sisteminiai procesai turi turėti aukštesnį prioritetą nei vartotojiški. Centrinio procesoriaus resursas yra vienintelis, dėl kurio varžosi tiek sisteminiai, tiek vartotojiški procesai. Modelyje vartotojiški procesai varžosi tarpusavyje (nes prioritetai neleidžia jiems varžytis su sisteminiais) tik dėl procesoriaus resurso.

# 1.9 Procesų primityvai

**Primityvas „kurti procesą“.**

Sukuriamas naujas deskriptorius masyve. Procedūra įvykdoma iššaukiančio proceso aplinkoje.

PROCEDURE KURTIP(n,s0,M0,R0,k0);

n – išorinis vardas

S0 – procesoriaus pradine būsena;

M0 – OA pradinė būsena (kiek išskirta OA resursu);

R0 – kiti išskiriami resursai;

K0 – proceso prioritetas

//duotu metu dirbantis procesas – „\*”

PROCEDURE KURTIP(n,s0,M0,R0,k0);

BEGIN

I:=NVV;(//NVV-naujas vidinis vardas); Id[i]:=n;CPU[i]:=s0; OA[i]:=M0; R[i]:=R0;//kitų

turimų proceso resursų komponente(nouroda į sąrašą); PR[i]:=k0;

ST[i]:=READYS;//laikysime, kad procesas sukuriamas su statusu(pvz. pasiruošes)

SD[i]:=PPS; T[i]:=\*;. S[i]:=Λ;//sūnų sąrašas iš pradžių jis tuščias Įrašyti(s[\*],i);//\*įgavo

naują sūnų i; Įjungti(PPS,i); END;

**Primityvas „naikinti procesą“**

Procesas naikina visus savo palikuonis (sūninius procesus), bet negali sunaikinti savęs. Tada jis nusiunčia pranešimą savo tėvui, kuris jį ir sunaikina. Pakartotino naudojimo resursus reikia atlaisvinti.

PROCEDURE NAIKINTIP(n);

BEGIN L:=false; jei L = true – yra vykdomų palikuonių, tai iškviečiamas planuotojas;

i:=VV(n); P:=T[i]; NUTRAUKTI(i); // panaikina visus proceso palikuonis

Pašalinti(S[T[i]], i); IF L THEN PLANUOTOJAS;END;

PROCEDURE NUTRAUKTI(i);

BEGIN

IF ST[i] = RUN THEN BEGIN STOP(i); L:=true; END;

Pašalinti(SD[i], i); // pašalinti iš pasiruošusių arba laukiančiųjų procesų sąr.

FOR ALL S є S[i] DO NUTRAUKTI(S);

FOR ALL r є OA[i] v R[i] DO

IF PNR THEN Įrašyti(PA[r], Pašalinti(OA[i] VR[i]));

FOR ALL r є SR[i] DO NDR(r); //naikinami resursų deskriptoriai

NPD(i); //naikiname proceso deskriptorių.END;

**Primityvas „Stabdyti procesą“.**

Tėvinis procesas gali stabdyti vaikinį nestabdant vaikinio palikuonių. n – išorinis vardas, a –adresas į proceso deskriptoriaus būseną (jo kopiją).

Procedure STABDYTIP(n,AR); Begin i:=vv(n); S:=ST[i];//einamasis proceso statusas//

if S=RUN then STOP(i);

[[[STOP(i) įvykdo veiksmus: 1) pertraukia procesorių P[i]; 2) įsimena pertraukto

procesoriaus P[i] būseną proceso deskriptoriaus komponentėje CPU[i] 3) PROC[P[i]]:=

Λ //pažymima, kad procesorius P[i] yra laisvas//]]]

AR:=copydest[i]; if S=BLOCK or BLOCKS then ST[i]:=BLOCKS else

ST[i]:=READYS; if s=RUN then PLANUOTOJAS; End

**Primityvas „aktyvuoti procesą“.**

Nuima pristabdymo buseną. Galbūt iškviečia planuotoją, jei būsena yra READY.

PROCEDURE AKTYVUOTI(n);

BEGIN i:=VV(n); ST[i]:=IF ST[i]=READYS THEN REDY ELSE BLOCK;

IF ST[i]=READY THEN PLANUOTOJAS; END;

**Primityvas „keisti proceso prioritetą“.**

Proceso įterpimas į sąrašą vyksta atsižvelgiant į proceso prioritetą, todėl proceso prioriteto pakeitimas vyksta taip: pašalinamas iš sąrašo ir po to įterpiamas pagal naują prioritetą.

PROCEDURE KEISTIPP(n,k);

BEGIN I:=VV(n); M:=PR[i]; Pašalinti(SD[i],i); PR[i]:=k; Įrašyti(SD[i],i); If M<k and

ST[i]=READY THEN PLANUOTOJAS; END;

**Primityvas „kurti resursą“**

Resurso deskriptoriaus sukūrimas. Resursus kuria tik procesas.

Procedure KURTIR (RS, PN, PAo, LPSo, PASKo);

RS – išorinis resurso vardas

PN - ar res.yra iš naujo panaudojamas

PA0 – res.prieinamumo aprašymas

LPS0 – to resurso laukiančių proc.sąr.

PASK0 – res.paskirstutojo programos adresas

Begin

r:=NRVV; Rid[r]:=RS; PNR[r]:=PN; k[r]:=\*; PA[r]:=PAo; LPS[r]:=LPSo;

PASK[r]:=PASKo; Įrašyti (SR[\*], r);End;

**Primityvas „naikinti resursą“**

Sunaikinti resursa gali jo tevas arba pirmtakas. Sunaikinamas resurso deskriptorius.

PROCEDURE NAIKINTIR(RS);

Begin

r:=RVV(RS); //atblokuojami resurso laukiantys proc.jiems pasiunčiant fiktyvų res.//

R:=Pasalinti(LPS[r]); While R<>1 do Begin ST[R.P]:=IF ST[R.P]=BLOCK then READ

else READS; Irasyti (PPS,R.P); SD[R.P]:=PPS; R.A:=’PRANEŠIMAS’;

R:=Pasalinti(LPS[r]); End

NRD(r); //Naikinri resurso deskriptoriu//; PLANUOTOJAS; End

**Primityvas „kurti resursą“.**

Resurso deskriptoriaus sukūrimas. Resursus kuria tik procesas.

Procedure KURTIR (RS, PN, PAo, LPSo, PASKo);

RS – išorinis resurso vardas

PN - ar res.yra iš naujo panaudojamas

PA0 – res.prieinamumo aprašymas

LPS0 – to resurso laukiančių proc.sąr.

PASK0 – res.paskirstutojo programos adresas

Begin

r:=NRVV; Rid[r]:=RS; PNR[r]:=PN; k[r]:=\*; PA[r]:=PAo; LPS[r]:=LPSo;

PASK[r]:=PASKo; Įrašyti (SR[\*], r);End;

**Primityvas „naikinti resursą“**

Sunaikinti resursa gali jo tevas arba pirmtakas. Sunaikinamas resurso deskriptorius.

PROCEDURE NAIKINTIR(RS);

Begin

r:=RVV(RS); //atblokuojami resurso laukiantys proc.jiems pasiunčiant fiktyvų res.//

R:=Pasalinti(LPS[r]); While R<>1 do Begin ST[R.P]:=IF ST[R.P]=BLOCK then READ

else READS; Irasyti (PPS,R.P); SD[R.P]:=PPS; R.A:=’PRANEŠIMAS’;

R:=Pasalinti(LPS[r]); End

NRD(r); //Naikinri resurso deskriptoriu//; PLANUOTOJAS; End

**Procesas StartStop**

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

Sisteminių resursų inicializacija

Sisteminių permanentinių procesų inicializacija

Blokavimasis laikiant „MOS pabaiga“ resurso

Sisteminių procesų naikinimas

Sisteminių resursų naikinimas

Start

Stop

**Procesas ReadFromInterface**

ReadFromInterface – užduoties nuskaitymo iš įvedimo srauto procesas.

Blokavimas laukiant „Iš vartotojo sasajos“ resurso

Duomenų nuskaitymas

Blokavimasis laukiant supervizorinės atminties resurso

Duomenų įrašymas į supervizorinę atmintį

Atlaivinamas resursas „Užduotis supervizorinėje atmintyje“. Skirta procesui JCL

**Procesas JobToSwap**

Šio proceso paskirtis – perkelti užduoties programos bloką iš vartotojo atminties į išorinę.

Blokavimasis laukiant „Užduoties vykdymo parametrai supervizorinėje atmintyje“ resurso

Blokavimasis laukiant „Išorinė atmintis“ resurso

Blokavimas laukiant „Kanalų įrenginio“ resurso

Įvykdoma duomenų apsikeitimo komanda

Atlaisvinamas „Kanalo įrenginys“ resursas

Atlaisvinamas „Užduotis išorinėje atmintyje“ resursas

**Procesas MainProc**

MainProc – Procesas valdantis JobGorvernor procesus, t.y., kuriantis užduočių vykdymo aplinkas ir jas naikinantis.

Blokavimas laukiant „Užduotis išorinėje atmintyje“ resurso

Naikinamas proc. JobGorvernor, sukūręs gautąjį res.

Kuriamas procesas JobGorvernor, suteikiant „Užduotis išorinėje atmintyje“ kaip pradinį res.

Vykdymo laikas ne 0 ?

NE

Taip

**Procesas Loader**

Loader – iš išorinės atminties užduoties programos tekstas perkeliamas į vartotojo atmintį.

Blokavimas laukiant „Pakrovimo paketo“ resurso

Blokavimas laukiannt „Kanalų įrenginio“ resurso

Nustatinėjami kanalų įrenginio registrai ir įvykdoma duomenų paėmimo komanda

Atlaisvinamas „Kanalų įrenginys“ resursas

Atlaisvinamas „Iš Loader“ resursas, skirtas JobGorvernor procesui sukūrusiam gautąjį „Pakrovimo paketo“ resursą.

**Procesas JobGorvernor**

JobGovernor – virtualios mašinos proceso tėvas, tvarkantis virtualios mašinos proceso darbą.

Blokavimasis laukiant „Vartotojo atmintis“ resurso

Atlaisvinamas „Pakrovimo paketas“ resursas

Blokavimasis laukiant „Iš Loader“ resurso

Atlaisvinamas „Išorinė atmintis“ resursas

Blokavimasis laukiant „Vartotojo atmintis“ resurso, skirto puslapių lentelei laikyti

Proceso „Virtuali mašina“ kūrimas

Blokavimasis laukiant „Iš Interrupt“ resurso

Proceso „Virtuali mašina“ stabdymas

SWAP pertraukimas?

Įvedimo / išvedimo pertraukimas pert.

iš

NE

TAIP

NE

Blk. laukiant atl. „Užduotis išorinėje atmintyje“ resurso

Ar įvedimas?

sukurti resursą „Užduoties vykdymo parametrai supervizorinėje atmintyje“ išorinėje atmintyje

TAIP

TAIP

Blokavimasis laukiant „Vartotojo įved.“ res. Šiam JobGorvernor

Naikinti proc. „Virtuali mašina“

NE

Atlaisvinti „Vartotojo atmintis“ res.

Sukurt resursą „eilutė“ supervizorinėje atmintyje.

Eilutės nuskaitymas į vartotojo atminties buferį.

Atlaisvinti fiktyvų „Užd .išor. atm.“ res.

Blokavimasis laukiant PrintLine proceso darbo pabaigos.

Blokavimasis laukiant „Neegzistuojantis“ res.

Proc. „Virtuali maš.“ aktyvavimas

**Procesas JCL**

JCL – procesas, interpretuojantis užduoties tekstą, išskiriantis programą, duomenis, parametrus ir užfiksuojantis užduoties formulavimo klaidas.

Blokavimas laukiant “užduotis supervizorinėje atmintyje“ resurso

Atlaisvinti „Eilutė atmintyje“ resursą su pranešimu apie atminties trūkumą VM programoje

Imamas blokas iš supervizorinės atminties

Programos įrašo inicijavimas

Ar tai $AMJ blokas?

Ar programų sąrašas netuščias?

Atlaisvinti “eilutė atmintyje“ resursą su pranešimu apie $AMJ bloko nekorektiškumą VM programoje ir atminties tipu „supervizorinė“

Atlaisvinti “eilutė atmintyje“ resursą su pranešimu apie $END bloko nebuvimą VM programoje ir atminties tipu „supervizorinė“

Atlaisvinamas „užduoties programa supervizorinėje atmintyje“ resursas

Atlaisvinamas „užduoties programa supervizorinėje atmintyje“ resursas

Atlaisvinamas „užduoties vykdymo parametrai supervizorinėje atmintyje“ resursas

Imamas blokas iš supervizorinės atminties

Ar tai $END blokas arba nėra blokų?

Ar $AMJ blokas korektiškas?

Ar tai $END blokas?

Prijungiamas blokas prie programos sąrašo

Atlaisvinti „eilutė atmintyje“ resursą su pranešimu apie programos nebuvimą VM programoje ir atminties tipu „supervizorinė“

Taip

Taip

Taip

Taip

Taip

Ne

Ne

Ne

Ne

**Procesas Interrupt**

Interrupt – procesas, apdorojantis virtualios mašinos pertraukimą sukėlusią situaciją.

Blokavimasis laukiant „pertraukimas“ resurso

Pertraukimo tipo identifikavimas

JobGorvernor, atsakingo už pertrauktosios VM darbą, nustatymas

Atlaisvinamas „iš interrupt“ resursas, skirtas konkrečiam JobGorvernor

**Procesas PrintLine**

Blokavimasis laukiant „Eilutė atmintyje“ resurso

Blokavimasis laukiant „Kanalų įrenginio“ resurso

Nustatinėjami kanalų įrenginio reg. ir įvykdoma duomenų nuskaitymo komanda

Atlaisvinamas „Kanalų įrenginys“ resursas

PrintLine – į išvedimo įrenginį pasiunčiama eilutė iš supervizorinės atminties.

**Procesas VirtualMachine**

Virtual Machine – procesas, atsakantis už vartotojiškos programos vykdymą.

Procesoriaus perjungimas į vartotojo režimą

Virtualios mašinos procesoriaus komandų emuliavimas

Sukuriamas pertraukimo įvykio resursas

# 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.

# Resurso deskriptorius

Resurso deskriptorius yra fiksuotoformato duomenų struktūra, sauganti informaciją apie resurso einamąjį stovį. Remiantis informacija resurso deskriptoriuje nurodomas jo užimtumo laipsnis, laisvas kiekis, nuoroda į pačius resurso elementus ir kt. Šia informacija naudojasi duotojo resurso paskirstytojas. Resurso inicializavimas reiškia deskriptoriaus sukūrimą. Darbas su deskriptoriais galimas tik per specialias operacijas - OS branduolio primityvus.

*r-resurso vidinis vardas (indeksas resursų deskriptorių masyve)*

1. Identifikacija;
   1. Rid[r] – išorinis vardas;
   2. PNR[r] – ar tai pakartotinio panaudojimo resursas (jei taip - jį reikia gąžinti);
   3. K[r] – resursą sukūrusio proceso vidinis vardas;
2. PA[r] – nuoroda į prieinamumo aprašymo sąrašo pradžią;
3. LPS[r] – nuoroda į resurso laukiančių procesų sąrašo prad;
4. PASK[r] – resurso paskirstytojo programos adresas.

# 2.3 Resurso primityvai

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.

Resurso deskriptoriaus sukūrimas. Resursus kuria tik procesas.

*Procedure KURTIR (RS, PN, PAo, LPSo, PASKo);*

*RS – išorinis resurso vardas*

*PN - ar res.yra iš naujo panaudojamas*

*PA0 – res.prieinamumo aprašymas*

*LPS0 – to resurso laukiančių proc.sąr.*

*PASK0 – res.paskirstutojo programos adresas*

*Begin*

*r:=NRVV; Rid[r]:=RS; PNR[r]:=PN; k[r]:=\*; PA[r]:=PAo; LPS[r]:=LPSo;*

*PASK[r]:=PASKo; Įrašyti (SR[\*], r);End;*

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.

Sunaikinti resursa gali jo tevas arba pirmtakas. Sunaikinamas resurso deskriptorius.

PROCEDURE NAIKINTIR(RS);

Begin

r:=RVV(RS); //atblokuojami resurso laukiantys proc.jiems pasiunčiant fiktyvų res.//

R:=Pasalinti(LPS[r]); While R<>1 do Begin ST[R.P]:=IF ST[R.P]=BLOCK then READ

else READS; Irasyti (PPS,R.P); SD[R.P]:=PPS; R.A:=’PRANEŠIMAS’;

R:=Pasalinti(LPS[r]); End

NRD(r); //Naikinti resurso deskriptoriu//; PLANUOTOJAS; End

**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.

Procesas, kuriam reikia resurso, iškviečia šį primityvą, nurodydamas išorinį vardą ir adresą. Toks procesas pereina į blokavimosi būseną. Blokavimasis įvyksta tik prašant resusrso. Procesas įjungiamas į laukiančių to resuro procesų sąrašą.

PROCEDURE PRASYTIR(RS, D, A);

D – kokios resusrso dalies prašoma:

A – atsakymo srities adresas, į kur pranešti.

BEGIN

R := RVV(RS); IJUNGTI(LPS[r], (\*, D, A)); PASK(r, K, L); // K – kiek procesų

aptarnauti, L – aptarnautų procesų vidinių vardų masyvas// B := true; //ar masyve L yra

einamas procesas// FOR J :=1 STEP 1 UNTIL K DO IF L[J] <> \* THEN BEGIN

i:=L[J]; Irašyti(PPS, i); SD[i]:=PPS; ST[i]:= IF ST[i]=BLOCK THEN READY ELSE

READYS; END ELSE B:=false; //(Procesas \* iš karto gavo resursą ir nebuvo įrašytas į

laukiančių procesų sąr.) IF B THEN

BEGIN

ST[\*]:=BLOCK;

SD[\*]:=LPS[r];

PROC[P[i]]:= Λ; //(Procesorius laisvas)

PASALINTI(PPS, \*)

END

PLANUOTOJAS

END

**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.

Tai atitinka situaciją, kai procesas gauna pakartotino naudojimo resursą ir kai jo jam nereikia, jis jį atlaisvina ir įjungia į sąrašą laisvųjų resursų bei papildo resurso prieinamumo aprašymo sąr.

PROCEDURE ATLAISVINTIR(RS,D);

BEGIN

r:=RVV(RS);

Ijungti(PA[r],D);//

PASK(r,k,L);

IF k>0 THEN FOR J:= 1 STEP 1 UNTIL k DO

BEGIN

i:=L(J);

Ijugti(PPS,i);

SD[i]:=PPS;

ST[i]:= IF ST[i]= BLOCKS THEN READYS

ELSE READY

END;

IF k<>0 THEN PLANUOTOJAS;

END;