VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Operacinės sistemos

3-asis LABORATORINIS DARBAS

„Multiprograminės virtualios ir realios mašinos projektas“

Atliko: Programų sistemų 2 kurso, 4 grupės studentas

Benediktas Gricius

VILNIUS
2013

Turinys

1 Įgyvendinama užduotis

4. Projektuojama interaktyvi OS.

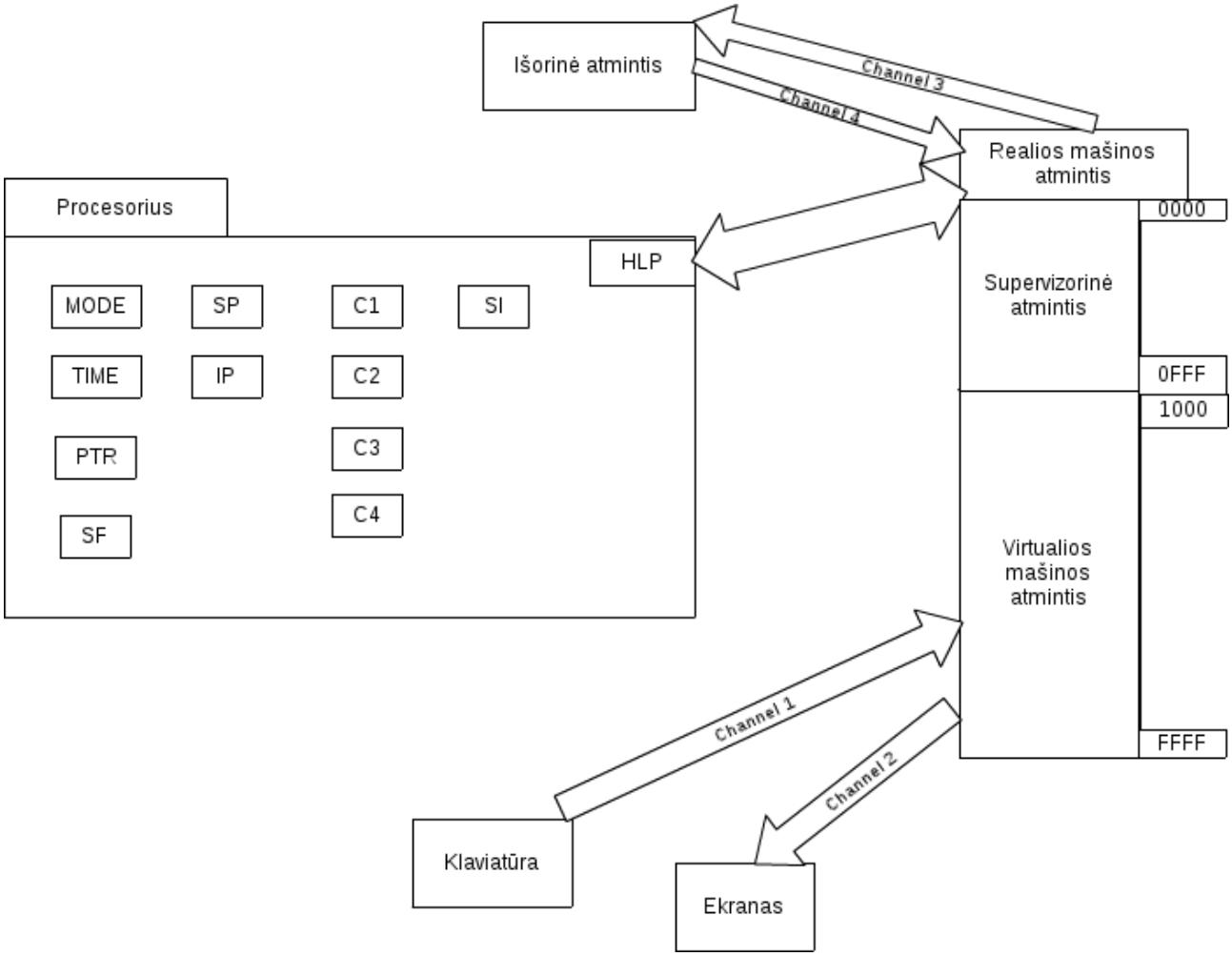
Virtualios mašinos procesoriaus komandos operuoja su duomenimis, esančiais steko viršūnėje. Yra komandos duomenų persiuntimui iš atminties į steką ir atvirkščiai, aritmetinės (sudėties, atimties, daugybos, dalybos), sąlyginio ir besąlyginio valdymo perdavimo, įvedimo, išvedimo ir programos pabaigos komandos. 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.

2 Realios mašinos aprašas

2.1 Realios mašinos schema



2.2 Realios mašinos registrai

Registras	Pavadinimas	Dydis bitais	Aprašymas
MODE	Realios mašinos rėžimo registras	8	Ne 0 – jei dirbama vartotojo rėžimu 0 – dirbama supervizoriniu rėžimu
PTR	Puslapių transliacijos registras	16	Vyresnysis baitas saugo puslapių lentelės pradžios bloko numerį. Jaunesnysis baitas saugo įrašų kiekį puslapių lentelėje.
SP	Stack Pointer (steko viršūnės registras)	8	Virtualios mašinos steko registras. Pradinė reikšmė 00.
A	Darbinis	32	Darbinis žodžio dydžio registras
IP	Instruction Pointer (komandų skaitliukas)	8	Virtualios mašinos komandų skaitliukas
SF	Status Flag (požymių registras)	8	Parodo procesoriaus būseną, vėliausio aritmetinio veiksmo rezultato požymis. **X** **X** **X** **X** **X** **TF** **ZF** **CF** X – nenaudojamas **TF** – Trap Flag – ar įjungtas debug rėžimas (taip → 1) **ZF** – Zero Flag – neįskaitant Carry bito visi rezultato bitai yra nuliniai. **CF** – Carry Flag – rezultatas netilpo skaičiaus be ženklo rėžiuose.
TIME	Timer	16	Skaičiuoja, kiek taktų liko iki sekančio pertraukimo signalo iškvietimo.
SI	Supervisor Interrupt	8	Parodo kilusio supervizorinio pertraukimo numerį.
PI	Program Inerrupt	8	Parodo kilusio programinio pertraukimo numerį.
C1	Channel 1	8	Ar šiuo metu yra atliekamas įvedimas klaviatūra? 0 – ne 1 – taip
C2	Channel 2	8	Ar šiuo metu yra atliekamas išvedimas į ekraną? 0 – ne 1 – taip
C3	Channel 3	8	Ar šiuo metu yra atliekamas rašymas į išorinę atmintį? 0 – ne 1 – taip
C4	Channel 4	8	Ar šiuo metu yra atliekamas skaitymas iš išorinės atminties? 0 – ne 1 – taip

2.3 Supervizoriniai pertraukimai

Pertraukimai į kuriuos reaguoja reali mašina. Apie tai, koks pertraukimas kilo reali mašina žino pagal SI registro reikšmę:

SI reikšmė	Pavadinimas	Aprašymas
0	-	Jokio pertraukimo
1	Full Stack	Jeigu reikia dalį steko atminties iškelti į swap atmintį. Kyla, kai SP reikšmė PUSH veiksmu padaroma į FF.
2	Empty Stack	Jei išsemtas visas stekas.
3	General protection	Peržengti segmento rėžiai (bandoma prieiti prie ne savo atminties)

Taip pat, nors atskiro pertraukimo numerio neišskiriama, bet po kiekvienos komandos įvykdymo patikrinama ar SF registre požymis TF yra aktyvuotas (bitas vienetas). Jei taip, yra įvykdoma žingsnio pertraukimo apdorojimo procedūra. Patogi naudoti debug rėžimui.

2.4 Programiniai pertraukimai

Virtualios mašinos vykdymo metu sukeliami pertraukimai. Apie kilusį pertraukimą sužinoma iš PI registro reikšmės.

PI reikšmė	Pavadinimas	Aprašymas
0	-	Jokio pertraukimo
1	Halt	Programa baigė darbą
2	OPK undefined	Neteisingas operacijos kodas
3	Operand undefined	Netinkamai apibrėžtas komandos operandas
4	Input	Reikia laukti įvedimo iš klaviatūros
5	Output	Reikia atlikti išvedimą į ekraną

2.5 Realios mašinos rėžimai

Reali mašina palaiko du vykdymo rėžimus:

- **Supervizorinį** – prieinama visi atmintis bei visi realios mašinos registrai, valdoma visa reali mašina.
- **Vartotojo** – vykdoma virtuali mašina, tačiau prieinama tik jai numatyta atminties sritis (kodas, duomenų segmentas ir stekas).

2.6 Puslapių transliacija

Realioje mašinoje numatyta puslapių transliacija.

Veikiant virtualiai mašinai PTR registre saugoma informacija:

- puslapių lentelės pradžios bloko numeris
- įrašų (eilučių) puslapių lentelėje kiekis.

Reali mašina norėdama sužinoti, kur yra kažkuris atminties puslapis ar žodis kreipiasi į puslapių lentelę ir pagal tai apskaičiuojamas to puslapio ar žodžio absoliutus adresas realioje mašinoje.

Kiekvienas įrašas puslapių lentelėje užima vieną žodį.

Pavyzdys:

Reali mašina turi 256 puslapius (blokus), kiekvieno dydis 256 baitai.

Pirmi 16 puslapių skiriami supervizorinės atminties reikmėms.

Paleidus programą prog.pr jos puslapiai atsitiktinai sudedami į laisvus realios mašinos puslapius.

Tarkim gavome tokį išdėstymą puslapiais: Stack(#20), Dataseg(#51), Codeseg(#31), PTT(#68).

Reiškia: 20-tas puslapis skiriamas stekui, 51 – duomenų segmentui, 31 – kodo segmentui,

Puslapių lentelė bus 68-tame puslapyje ji susidės iš 3 žodžių: 20,51,31.

2.7 Swapping'o palaikymas

Vykdomos virtualios mašinos turi tik kodą ir steką (šioje realizacijoje nebus naudojamas duomenų segmentas).

Programoms operuojant steke esančiais duomenimis kyla galimybė steke nesutalpinti visų reikalingų duomenų. Todėl yra numatytas swapping mechanizmas.

Kiekvienai virtualiai mašinai sukuriamas atskiras laikinas failas swappingo mechanizmui vykdyti. Jis yra ištrinamas, kai virtuali mašina įvykdo komandą HALT.

Pasitelkiant swappingą, kai stekas yra užpildomas pilnai, tada pusė jo žodžių yra iškeliami į išorinę atmintį. Išimant iš steko elementus ir pagrindinėje atmintyje nebelikus steko žodžių, bet kai jų yra išorinėje atmintyje naudojamoje swappingui, jie paimami iš šios išorinės atminties ir sukeliami į steką.

Taip vartotojui (virtualiai mašinai) sudaromas įspūdis, kad stekas yra didelis, nors realiai jam skiriamas tik vienas atminties blokas.

2.8 Taimerio mechanizmas

Reali mašina turi registrą TIME.

Po kiekvienos įvykdytos komandos šio registro reikšmė yra sumažinama vienetu. Pradinę reikšmę gali nusistatyti vartotojas dirbantis su operacine sistema suvedęs komandą:

Numatytoji reikšmė paleidus virtualią mašiną yra 10.

Viena komanda laikoma vienu procesoriaus taktu, o išvedimo ir įvedimo – dviem. Vos TIME registro reikšmė yra 0, kyla supervizorinis taimerio pertraukimas: įvykdoma timerio pertraukimo apdorojimo procedūra ir TIME reikšmė vėl nustatoma į 10.

Kilus šiam pertraukimui planuotojas iš vykdomų procesų sąrašo išrenka sekantį ir imasi vykdyti jo komandas.

2.9 Įvedimo ir išvedimo įrenginiai

Projektuojama OS yra interaktyvi. Ji leidžia vartotojui suvedant komandas reguliuoti realios mašinos veikimą, bei matyti tų veiksmų rezultatus.

Įvedimo įrenginys – klaviatūra, atitinkamą komandą vartotojas suveda kaip simbolių eilutę ir norėdamas vykdyti spaudžia ENTER klavišą.

Išvedimo įrenginys – ekranas, vartotojui realios mašinos darbo rezultatas parodomas ekrane.

2.10 Atminties įrenginiai

Išorinė atmintis – kietasis diskas, atminties tipas, kur duomenys išlieka ir nedirbant realiai mašinai.

Realios mašinos atmintis – tai atminties erdvė, kurioje yra 256 blokai po 256 žodžius. Žodžio dydis yra 32 bitai. Ši skirstoma į supervizorinę ir virtualios mašinos atmintis.

Supervizorinė atmintis - Realios mašinos atminties pradžios 16 blokų naudojami tik realios mašinos reikmėms ir virtualiai mašinai yra neprieinami. Ten gali būti saugomi realios mašinos darbui reikalingi duomenys.

Virtualios mašinos atmintis – Keletas puslapių realios mašinos atminties, kurie skiriami virtualios mašinos reikmėms – kodui ir stekui. Pritrūkus jos yra naudojamasi swappingo galimybe.

2.11 Supervizorinė atmintis

0-tajame puslapyje (bloke) yra saugoma informacija, kurie puslapiai yra užimti, o kurie laisvi. I-tasis puslapio žodis nurodo:

0 – puslapis laisvas

-**1** – puslapis užimtas

3 Virtualios mašinos aprašas

3.1 Virtualios mašinos vykdymas

Prireikus realiai mašinai vykdyti realios mašinos užduotį yra išskiriama atmintis:

- Vienas blokas puslapių lentelei
- Vienas blokas kodo segmentui
- Vienas blokas steko segmentui (esant poreikiui naudojamas swapping'as, dalis informacijos padedama į failą)

Yra vykdomas kodas esantis kodo segmente ir operuojama tik tais duomenimis, kurie yra steko viršūnėje.

3.2 Virtualios mašinos registrai

SP – stack pointer – 1 baito steko viršūnės poslinkis nuo steko puslapio pradžios

IP – instruction pointer – 1baito virtualios mašinos komandų skaitliukas, poslinkis nuo kodo puslapio pradžios.

Vykdant komandas nuosekliai IP rodo į šiuo metu vykdomą kodo eilutę. Steko viršūnės registras rodo į vėliausiai steke padėtą elementą.

3.3 Virtualios mašinos atmintis

Virtualios mašinos atmintis yra sudaryta iš 256 blokų po 256 žodžių.

Žodžio ilgis – 4 baitai (32 bitai).

Komandos dirba su steke esančiais duomenimis. Valdymo perdavimo komandos reaguoja į SF registro požymius.

3.4 Virtualios mašinos komandų sistema

3.4.1 Ne steko komandos

HALT – baigti virtualios mašinos darbą

3.4.2 Duomenų persiuntimas į steką

PNxx – PUSH Number (xx – baito šešioliktainėje sistemoje reikšmė 00-FF)

PCPY – PUSH Copy – į steką dar kartą padedama steko viršūnės reikšmė

3.4.3 Steko viršūnės peržiūra ir valdymas

POPx – POP Nowhere išimama steko viršūnės reikšmė, bet niekur nepadedama

TOPN – Steko viršūnės reikšmė išspausdinama kaip skaičius ekrane jos neišimant

TOPS – Steko viršūnės reikšmė išspausdinama kaip simbolis ekrane jos neišimant

POPS – Steko viršūnės reikšmė išimama iš steko ir išspausdinama ekrane

SWAP – Steko viršūnės ir už jos einančio žodžių reikšmės sukeičiamos

POPA – Steko viršūnės reikšmė įdedama į darbinį registrą

PUSA – Darbinio registro reikšmė padedama į steką

PUSW – Į steką padedamas žodis iš duomenų segmento (kokiu poslinkiu parodo darbinis registras A)

POPW – Į duomenų segmentą iš steko paimamas žodis (kokiu poslinkiu parodo darbinis regisras A)

3.4.4 Įvedimo ir išvedimo komandos

PRNL – Išspausdinamas naujos eilutės simbolis

PRNT – (PRiNT) Iškviečiamas pertraukimas simbolių eilutei išspausdinti (A reikšmė rodo kiek simbolių, steko viršūnės reikšmė – nuo kurio).

PRSx – Išspausdinamas vietoj x nurodytas simbolis

3.4.5 Aritmetinės virtualios mašinos komandos

Atliekant aritmetinis operacijas yra išimami abu operandai iš steko, suskaičiuojamas rezultatas ir šis patalpinamas į steką. Naudojant jas yra nustatomi SF požymiai (flagai) CF ir ZF.

MULT – Steko viršūnėje esančių reikšmių daugyba

ADDT – Steko viršūnėje esančių reikšmių sudėtis

SUBT – Steko viršūnėje esančių reikšmių atimtis (iš reikšmės esančiossteko viršūnėje atitmama reikšmė esanti už jos)

DIVT – Steko viršūnėje esančių reikšmių dalyba. Už steko viršūnės esanti reikšmė padalinama iš steko viršūnės reikšmės. Kaip rezultatas į steką paeiliui padedami: rezultatas, po to liekana.

CMPT – Atliekama SUBT nekeičiant steko būsenos (palyginimo operacija, nustatomi tik SF požymiai).

3.4.6 Valdymo perdavimo komandos

Besąlyginis valdymo perdavimas:

JMxx – pereinama vykdyti kodo eilutės numeriu xx (xx – du šešioliktainiai skaitmenys, t. y. vieno baito skaičius).

Sąlyginis valdymo perdavimas:

Patikrinama sąlyga. Jei ji netenkinama, einama vykdyti sekančios komandos paeiliui. Jei tenkinama – einama vykdyti komandos eilutėje xx, kur xx – vieno baito reikšmė išreiškta dviem šešioliktainiais skaitmenimis.

(skirta naudoti po CMPT komandos).

Komanda	Pavadinimas	Tikrinama sąlyga (SF registre)
JExx	Jump If Equal	ZF=1
JNxx	Jump if Not Equal	ZF=0
JAxx	Jump if Above	CF=0
JBxx	Jump if Below	CF=1
JaXX	Jump if Above or Equal	CF=0
JbXX	Jump if Below or Equal	CF=1 arba ZF=1

3.5 Virtualios mašinos užduočių formulavimas

Užduoties kodas yra surašomas faile. Visame faile rašant tik vykdymui reikalingas komandas. Paskutinė vykdoma komanda privalo būti HALT. Kitu atveju virtualios mašinos darbo rezultatai nėra apibrėžti.

Bendra programos schema:

#DATASEG:

DW simbolinis žodis – 4simboliai tarp kabučių (pvz.: DW "ABCD")

DPI neneigiamas sveikas skaičius (pvz. DPI 5)

DNI neigiamas sveikas skaičius (pvz. DNI -2)

#CODESEG:

<komandos po 4 simbolius>

HALT

Pastaba:

Naują eilutę žymi žodis, kurio skaitinė reikšmė -1 (t. y. DNI -1)

Eilutės pradžioje panaudotas // **reiškia komentarą** (eilutė praleidžiama ir neparsinama jos tolesnė sintaksė). Tuščios arba tarpų pilnos eilutės taip pat praleidžiamos.

Prieš komandą ir po jos tarpai leidžiami.

3.5.1 „Labas Pasauli“ programa

#DATASEG:

DW 'Laba'

DW 's pa'

DW 'saul'

DW 'i---'

#CODESEG:

PU0D

//darbinis registras parodys spausdinamo stringo ilgį

POPA

//steko viršūnėje – poslinkis nuo duomenų segmento pradžios

PU00

//iškviečiamas pertraukimas simbolių eilutei išspausdinti

PRNT

//išspausdinamas \n

PRNL

HALT

4 Procesai ir jų primityvai

4.1 Procesų būsenos

READY – Pasiruošęs, laukiantis procesoriaus vykdymui.

BLOCK – Užsiblokavęs, laukiantis kokio nors resurso, kad galėtų tęsti savo darbą.

RUN – Šiuo metu vykdomas procesas.

Vos sukurtas vartotojo procesas pakraunamas atminty ir patalpinamas į laukiančių READY procesų eilę (sisteminiai visi pažymimi kaip BLOCKED).

READY → RUN: Procesui paskirtas procesoriaus laikas, proceso vykdymas tęsiamas.

READY → BLOCK: Procesas pareikalavo resurso, kurio dar nėra sukurta. Pereina į resurso laukimo rėžimą. Neatliks jokių veiksmų, kol negaus atitinkamo resurso (įmanoma, kad jis to resurso nesulauks niekada).

BLOCK → READY: Procesas gavo reikalautą resursą. Jis gali būti vykdomas toliau, tereikia sulaukti, kol procesorius imsis jį vykdyti. Procesas patalpinamas į pasiruošusių procesų eilės pradžią.

4.2 Proceso deskriptorius

T.y. duomenų struktūra laikanti visą informaciją, kurią operacinei sistemai būtina žinoti apie procesą, kad galėtų juos vykdyti paeiliui ir naudotų tinkamus resursus.

PID – Process ID – Unikalus proceso vardas.

State – Einamojo proceso būsena (Ready,Blocked,Running)

PPID – Process parent ID – Proceso tėvo ID.

PTT – Page translation table – Puslapio numeris atminty, kur yra šio proceso puslapių transliacijos lentelė.

RES – Sąrašas laukiamų resursų (aktualus tik procesui esant užsiblokavusiam). Jei procesas yra READY arba RUN, šis sąrašas yra tuščia.

4.3 Procesų sąrašai

Sistemoje veikia **sisteminiai** ir **vartotojo** procesai. Sisteminiai procesai turi aukštesnį prioritetą. Sudaromi 4 dvipusiai cikliniai sąrašai (nuo aukščiausio prioriteto):

1. Sisteminiai BLOCK
2. Sisteminiai READY
3. Vartotojo BLOCK
4. Vartotojo READY

4.4 Sisteminiai procesai

Pradėjus veikti operacinei sistemai sukuriamas procesas Init ir jo vaikai.

PID	Proceso vardas	Paskirtis
1	Init	Visų procesų tėvas (nuolat užsiblokavęs, kol dirba OS).
2	MemManager	Operatyviosios atminties valdymui
3	FileManager	Išorinės atminties (kietojo disko) valdymui
4	StdInputManager	Įvedimo srautui kontroliuoti
5	StdOutputManager	Išvedimo srautui kontroliuoti
6	ProgramLoader	Vartotojo programos pakrovimui į atmintį
7	ProgramKiller	Atlaisvina programos valdytą atmintį, resursus ir pašalina ją iš procesų sąrašo.
8	ResManager	Resursų paskirstytojo procesas

4.4.1 Init

Visi procesai sistemoje yra šio proceso vaikai arba įpėdiniai. Vos sukurtas yra užsiblokavęs ir tik gavęs „OSExit“ resursą aktyvuojasi ir iškart baigia darbą. Sukuriamas pats pirmas sistemoje. Pats paskutinis sistemoje baigia darbą.

4.4.2 MemManager

Procesams prireikus darbo su atmintimi kreipiamąsi į MemManager procesą sukuriant resursą „MemoryAccess“. Pagal gautus parametrus šis procesas apdoroja prašymą, taip pat užtikrina, kad nebūtų pažeista General Protection taisyklė. Su šiuo procesu intensyviai bendradarbiauja visi procesai (ypač sisteminiai). Jis yra atsakingas už tinkamą atminties panaudojimą.

4.4.3 FileManager

Failų sistemos valdymo procesas. Gavęs failų sistemos kontrolės resursą, konkrečiai nuo to kokį gavo: sukuria, keičia, trina atitinkamus failus nurodytose particijose. Esant reikalui generuoja klaidų pranešimus.

4.4.4 StdInputManager

Atsiradus resursui „NeedInput“ atsiblokuoja. Priima OS naudotojo įvestus duomenis, juos perduoda jų prašiusiam procesui ir vėl užsiblokuoja.

4.4.5 StdOutputManager

Atsiradus resursui „NeedOutput“ atsiblokuoja. Atlieka reikalingą išvedimą ir vėl užsiblokuoja.

4.4.6 ProgramLoader

Gavęs pakraunamos programos resursą, sukuria jos puslapių lentelę, rezervuoja puslapius kodo, steko, duomenų segmentams, įtraukią procesą į pasiruošusių eilę.

4.4.7 ProgramKiller

Sukūrus resursą „KillProcess“ atlieka priešingus veiksmus nei ProgramLoader. Atlaisvina operatyvios atminties puslapių lentelę, segmentams skirtus puslapius, swapą (ištrina einamosios programos swapui naudotą failą).

4.4.8 ResManager

Procesas, bandantis paskirstyti resursus jų laukiantiems procesams.

4.5 Procesų planuotojas (Scheduler)

Persijungimą tarp procesų kontroliuoja planuotojas.

Įvykus laikrodžio pertraukimui imamas kitas sisteminis READY procesas, jei tokio nėra → vartotojo READY procesas ir vykdomas iki kito pertraukimo.

Procesai vartotojo ir sisteminiuose sąrašuose dirba atskirai (sąrašai neturi bendrų elementų).

Procesai parenkami „round-robin“ algoritmu.

Procesų valdymą planuotojas atlieka naudodamas procesų primityvus.

4.6 Procesų planuotojo darbe naudojami primityvai

4.6.1 Kurti procesą

Išskiriama atmintis naujo proceso vykdymui. Sukuriamas resursas programos pakrovimui į operatyviąją atmintį. Būtina sąlyga: turi būti laisvos atminties šiam procesui pakrauti. (atsiblokuos ProgramLoader procesas).

4.6.2 Aktyvuoti procesą

Nurodytas procesas gavęs resursą, kurio laukdamas užsiblokavo yra aktyvuojamas.

4.6.3 Blokuoti procesą

Nurodytas procesas paprašo resurso, kurio dar nėra resursų pool'e. Jis yra įrašomas į užblokuotų procesų eilę ir negali būti vykdomas, kol laukiamo resurso.

4.6.4 Naikinti procesą

Naikinamas egzistuojantis procesas. Įprastu atveju: vartotojo procesas, kuris baigė darbą komanda HALT. Tačiau gali būti ir sisteminių procesų darbo pabaiga OS darbo pabaigoje.

5 Resursų aprašas

Resursų sąvoka skirta tam, kad galėtume abstrakčiai apibrėžti bendravimą tarp procesų, užtikrinti sklandų multiprograminės operacinės sistemos darbą.

5.1 Resursų pool'as

Ciklinis tiesinis sąrašas, skirtas egzistuojantiems resursų deskriptoriams saugoti.

Norint sukurti resursą, privaloma pirmiau sukurti jo deskriptorių.

Gali būti, kad keli procesai bus sukūrę to paties tipo resursų.

5.2 Resursų deskriptorius

Duomenų struktūra laikanti informaciją apie konkretų resursą.

RID – Resource ID – Unikalus identifikatorius

Type – Resurso tipo magic number.

PID – Proceso, paprašiusio šio resurso ID.

DescState – Deskriptoriaus būsena:

- Tuščias (*Empty*)
- Resursas sukurtas ir naudojamas (*Active*)
- Resursas nebereikalingas (*Released*)

Content – Simbolių eilutė. Daugiausia vieno puslapio (256 baitų) dydžio. Saugojama specifinė resurso informacija. Pvz.: spausdinama/įvesta simbolių eilutė, atidarymui reikalingo failo vardas, ir pan.

5.3 Resursų tipai ir magic number

Skliaustuose įrašyti procesai nėra sisteminiai procesai, bet OS dalys.

(INT) reiškia, kad įvykus kažkokiam programiniam/supervizoriniam, bet ne laikrodžio pertraukimui kyla poreikis sukurti tą resursą.

(USER) reiškia kokį nors vartotojišką procesą

MagicNumber – sutartinis skaičius, resurso tipui atpažinti.

Pavadinimas – resurso pavadinimas sistemos terminais

Laukiantis procesas – procesas, kuris laukia tokio resurso.

Kuriantis procesas – procesas, kuris kuria tokio tipo resursus

„kažkur kitur atmintyje“ atitinka „parametrais nurodytoje atminties vietoje“

Magic Number	Pavadinimas	Laukiantis procesas	Kuriantis procesas	Aprašymas
1	OSExit	Init	ProgramKiller	Reikia baigti OS darbą
2	ReadBlock	MemManager	(bet koks procesas)	Nuskaitomas nurodyto dydžio blokas iš vienos atminties vietos į kitą (įrašomas į duomenų segmentą).
3	WriteBlock			Duomenų segmente esantis blokas įrašomas kažkur kitur atmintyje.
4	ReadWord			Vienas žodis iš atminties įrašomas į duomenų segmentą.
5	WriteWord			Vienas žodis iš duomenų segmento įrašomas kažkur atmintyje.
6	PushWord			Kadangi stekui prireikia swappingo mechanizmo, įsivedėme patogiam darbui su steku resursus (atskiriant įprastą darbą su duomenų segmentu, nuo darbo su steku).
7	PopWord			
8	NeedInput	StdInputManager	(INT)	Paimtas įvedimo srautas iš vartotojo
9	NeedOutput	StdOuputManager	(INT)	Išvedimui sukurtas simbolių srautas
10	ReadFile	FileManager	(INT)	Failo dalies nuskaitymas
11	OpenFile			Failo atidarymas (gauname jo deskriptorių)
12	WriteFile			Rašymas į failą su nurodytu poslinkiu
13	CloseFile			Failo uždarymas (paleidžiame jo deskriptorių)
14	DeleteFile			Failas nurodytu deskriptoriumi ištrinamas iš disko
15	SeekFile			Pakeičiama failo pozicija
16	ProgramStart	ProgramLoader	(INT)	Reikia pakrauti nurodytą programą į atmintį
17	ProgramHalt	ProgramKiller	(INT)	Reikia nutraukti nurodytos programos darbą
(bet kuris)	(bet koks resursas)	ResManager	(bet koks procesas)	Periodiškai paleidžiamas resursų paskirstytojas, kad būtų patikrinta ar koks nors resursas yra aktyvus.

5.4 Resursų paskirstytojas

Procesas „ResManager“. Naudodamasis kreipiniais į resursų primityvus, bei resursų pool'u, ieško galbūt kažkokiam procesui reikalingas sukurtas resursas.

Jei taip, tą procesą pažadina, perduoda procesui, duomenis apie resursą, kurių jam reikėjo.

Jei ne, pereina prie kito resurso sąraše ir vėl tikrina.

Jei pereitas visas resursų sąrašas, arba jau paskirtas kažkoks resursas, šis procesas atlieka pertraukimą, kuris vėl aktyvuoja planuotoją, o planuotojas toliau savo nuožiūra parenka procesus vykdymui.

5.5 Resursų paskirstytojo naudojami primityvai

Resursai valdomi per jų deskriptorius. Prieš prašant resurso būtina būti susikūrus deskriptorių tam resursui, o atlaisvinus resursą reikia sunaikinti ir jo deskriptorių. Taip leidžiama, keliems procesams prašyti tokio pat tipo resurso. Taip pat resursai savyje gali turėti specialių parametrų, perduodamų procesams.

5.5.1 Kurti deskriptorių

Reikės resurso, sukuriama duomenų struktūra informacijai apie procesą laikyti. (tuščia duomenų struktūra).

5.5.2 Prašyti resurso

Nurodytam deskriptoriui bandoma įvesti realias reikšmes prašant resurso. Jei toks resursas, yra jis suteikiamas. Jei ne, resursų paskirstytojas gauna pranešimą apie klaidą.

5.5.3 Atlaisvinti resursą

Panaudojus resursą jis pažymimamas, kaip panaudotas ir nebereikalingas. Jo duomenų struktūroje pažymima, kad resursas laisvas.

5.5.4 Naikinti deskriptorių

Atlaisvinus resursą, pašalinama apie jį informaciją saugojusi duomenų struktūra.

6 Failų sistemos principai

Apibrėžiama, išorinės atminties, virtualaus kietojo disko struktūra.

Kietasis diskas sudalinamas į vienodo dydžio sektorius (dalis, kurios arba naudojama visa, arba nenaudojama išvis). Vieno sektoriaus dydis šioje architektūroje sutampa su puslapio dydžiu: 256 žodžiai po 32 bitus.

16 sektorių sudaro takelį.

6.1 Pirmasis sektorius

Kietojo disko pirmas sektorius užpildytas reikšminga informacija apie patį diską:

Offset žodžiais	Dydis žodžiais	Prasmė
0	1	Kiek takelių sudaro kietąjį diską
1	1	Kiek įrašų yra particijų lentelėje
2	1	Žodis „HDD“ validumui patikrinti.
3i	1	i-osios particijos (skaičiuojant nuo 0) dydis takeliais.
3i+1	1	i-osios particijos paskirtis (0-failai, 1-swap'as)
3i+2	1	i-osios particijos vardas (max vieno žodžio, t. y. 4 baitų)

6.2 Particijos struktūra

Particiją sudaro **X** takelių. Pirmi X*16 (sektorių kiekis particijoje) baitų skiriami particijoje esančių **sektorių užimtumo žemėlapiui.**

Po šio žemėlapio sekančiame sektoriuje eina failų išsidėstymo lentelė. Joje yra Y įrašų:

* - Offset imant nuo sekančio sektoriaus particijoje po žemėlapio.

Offset žodžiais	Dydis žodžiais	Prasmė
0*	4	Failo vardas pvz. „duom.txt\0“ (baigiasi \0)
4	n	Failo užimamų sektorių sąrašas iš eilės
4+n	1	Failo dydis

Iškart po sektorių užimtumo žemėlapio ir failų sąrašo eina sektoriai su pačiais failų duomenimis.