Turinys

1. Įvadas

1.1. Skaičiavimo sistemos

Skaičiavimo sistema – tai būdų ir priemonių visuma, leidžianti užrašyti, ar kitaip pateikti skaičius skaitmenimis. Yra daugybė skaičių užrašymo skaitmenimis būdų. Bet kuri, pritaikyta praktiniam naudojimui skaičiavimo sistema turi užtikrinti:

- Galimybę užrašyti bet kurį skaičių iš nagrinėjamo skaičių diapazono;
- Užrašymo vienareikšmiškumas (kiekvienai simbolių kombinacijai turi atitikti vienas ir tik vienas dydis);
- Operavimo skaičiais paprastumą.

Visos skaičių pateikimo sistemos skirstomos į *pozicines* ir *nepozicines*.

Nepozicinėse skaičiavimo sistemose simbolio reikšmė nepriklauso nuo jo padėties skaičiuje. Tokių sistemų sudarymo principai nėra sudėtingi. Jų sudarymui naudojamos pagrinde sudėties ir atimties operacijos. Pavyzdžiui, sistema su vienu simboliu-pagaliuku buvo naudojama daugelyje tautų. Tam, kad užrašyti kažkokį tai skaičių šioje sistemoje, reikia užrašyti tam tikrą skaičių pagaliukų, lygų šiam skaičiui. Ši sistema nėra efektyvi, kadangi skaičiaus užrašymas tampa labai ilgas. Kitu nepozicinės sistemos pavyzdžiu gali būti romėnų sistema, kurios pagalba skaičiai užrašomi simbolių I, V, X, L, C, D ir kt. dėka. Šioje sistemoje nėra skaitmens nepriklausomybės nuo jo padėties skaičiuje. LX ir XL skaičiuose simbolis X įgauna dvi reikšmes: +10 ir -10.

Pozicinėje skaičiavimo sistemoje skaitmens reikšmė apibrėžiama pagal jo padėtį skaičiuje: vienas ir tas pats ženklas įgauna įvairias reikšmes. Pavyzdžiui, dešimtainiame skaičiuje 222 pirmas skaitmuo iš dešinės reiškia du vienetus, kitas – du dešimtis, o kairysis – du šimtus.

Dešimtainė skaičiavimo sistema labiausiai paplitusi skaičiavimo praktikoje. „Populiarumą“ ji įgijo ne kažkokiomis ypatingomis privilegijomis prieš kitas sistemas, o atsitiktinai – žmogaus rankos turi dešimt pirštų.

Bet kuri pozicinė skaičiavimo sistema charakterizuojama jos pagrindu. *Pozicinės skaičiavimo sistemos pagrindas (bazė)* – tai ženklų ar simbolių skaičius, naudojamas skaitmenims užrašyti duotoje sistemoje. Todėl gali egzistuoti begalė pozicinių skaičiavimo sistemų, kadangi už pagrindą galima imti bet kokį skaičių, taip sudarant naują sistemą. Pavyzdžiui, įmanoma šešioliktainė skaičiavimo sistema, kurioje skaičius užrašyti galima sekančių simbolių dėka: 0,1, ..., 9, A, B, C, D, E, F (vietoj A, ..., F galima užrašyti bet kurios kitus simbolius, pavyzdžiui, $\bar{1}, \bar{2}, ..., \bar{5}, \bar{6}$).

Pozicinėje skaičiavimo sistemoje galioja lygybė

$$A_{(q)}=a_{n-1}q^{n-1}+...+a_1q^1+a_0q^0+a_{-1}q^{-1}+...+a_{-m}q^{-m}. \quad (2)$$

Kur $A_{(q)}$ – bet koks skaičius, užrašytas skaičiavimo sistemoje, kurios pagrindas q; a_i – skaičiavimo sistemos skaitmenis; n, m – sveikų ir trupmeninių eilių skaičius.

Praktikoje naudojama sutrumpinta skaičių užrašymo forma:

$$A_{(q)}=a_{n-1}...a_1a_0a_{-1}...a_{-m}. \quad (3)$$

Taigi, sutrumpinta forma užrašytą skaičių 86,54, remiantis (2) galima užrašyti:
$86,54_{(10)}=8\cdot10^1+6\cdot10^0+5\cdot10^{-1}+4\cdot10^{-2}$.

Dvejetainėje skaičiavimo sistemoje skaičiams užrašyti naudojami du skaitmenis **0** ir **1**. Pagal (2) lygybę, dvejetainio skaičiaus (1001,1101) reikšmė:
$1001,1101_{(2)}=1\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0+1\cdot2^{-1}+1\cdot2^{-2}+0\cdot2^{-3}+1\cdot2^{-4}$.

Jeigu pagal dešimtainės sistemos aritmetikos taisykles atliksime veiksmus su dešinę aukščiau užrašytos lygybės puse, tai gausime dešimtainį duoto dvejetainio skaičiaus ($1001,1101_{(2)}$) ekvivalentą. Jis lygus $9,8125_{(10)}$.

Lentelėje pateikti kai kurie dešimtainės skaičiavimo sistemos skaičių ekvivalentai kitose skaičiavimo sistemose.

Lentelė 1

Dešimtainis skaičius	**Ekvivalentai kitose skaičiavimo sistemose**		
	q=2	**q=8**	**q=16**
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A

Tam, kad užrašyti vieną ir tą patį skaičių įvairiose skaičiavimo sistemose, reikalingas skirtingas pozicijų arba eilių kiekis. Pavyzdžiui, $96_{(10)}=140_{(8)}=1100000_{(2)}$. Kuo mažesnis sistemos pagrindas, tuo didesnis skaičiaus ilgis (skiltinio tinklelio ilgis).

Jeigu skiltinio tinklelio ilgis yra užduotas, tai riboja maksimalų pagal absoliutinę vertę skaičiaus reikšmę, kuri gali būti užrašyta.

Tegul skiltinio tinklelio ilgis lygus bet kuriam teigiamam skaičiui, pavyzdžiui, N. Tada

$$A_{(q)max}=q^N-1 \qquad (4)$$

Bet jeigu užduota maksimali absoliutinė skaičiaus reikšmė, tai skiltinio tinklelio ilgis

$$N=log_q(A_{(q)max}+1) \qquad (5)$$

Skaičių ašies intervalas, tarp maksimalaus ir minimalaus skaičių, vadinamas *skaičių vaizdavimo diapazonas (VD)* duotoje skaičiavimo sistemoje duotam skiltinio tinklelio ilgio apribojimui:

$$-A_{(q)max} \le VD \le A_{(q)max} \qquad (6)$$

Skaičiavimo mašinose apdorojamų skaičių ilgis paprastai apribotas tokiomis reikšmėmis: 1 baitas (8 skiltis), 2 baitai (16 skilčių), 4 baitai (32 skiltis), 8 baitai (64 skiltis). Atitinkamai yra riba ir užrašomiems nurodyto skiltinio tinklelio ilgio sveikiesiems skaičiams. Pavyzdžiui, maksimalus sveikas teigiamas skaičius, kuris gali būti užrašytas naudojant 16 dvejetainių skilčių, lygus $2^{16}-1=65535$.

Sistemos pagrindo santykis su skiltinio tinklelio ilgiu yra vadinamas sistemos ekonomiškumo rodikliu:

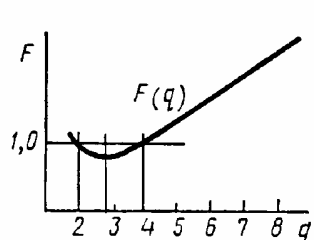
$$C=qN \qquad (7)$$

Tegul duota maksimali skaičiaus $A_{(q)max}$ reikšme. Tada ekonomiškumo rodiklis santykinai charakterizuoja tą įrangą, kuri reikalinga ESM atvaizduoti duotą skaičių įvairiose skaičiavimo sistemose.

Įstatant į (7) lygtį išreikštą per $A_{q(max)}$ N reikšmę (5), gausime

$$C_{(q)}=q\log_q(A_{(q)}+1) \qquad (8)$$

Paveikslėlyje pateiktas funkcijos $F(q)=C_{(q)}/C_{(2)}$ grafikas. J naudojant galima palyginti skaičiavimo sistemos, kurios pagrindas q ekonomiškumą su dvejetaine skaičiavimo sistema. Iš pateikto grafiko matome, kad iš visų galimų skaičiavimo sistemų su sveikuoju pagrindu ekonomiškiausia yra trejetainė skaičiavimo sistema.



Pasirodė, kad naudoti trejetainę skaičiavimo sistemą skaičiavimo mašinose sudėtinga dėl per daug komplikuotų elektroninių schemų. Dvejetainė skaičiavimo sistema, truputį atsiliekanti pagal ekonomiškumą, tapo daug populiaresnę skaičiavimo mašinose ir lengvesnė realizuoti.

Dvejetainės skaičiavimo sistemos panaudojimas ESM susietas dar ir su papildomais sunkumais, kai reikia paversti įvedamus į ESM skaičius į dvejetainę skaičiavimo sistemą ir atvirkščiai jas išvedant iš ESM. Tokie skaičių „virsmai“ atliekami automatiškai panaudojant specialiai sukurtus metodus.

Tam, kad paversti sveikuosius skaičius ir sveikąsias netaisyklingų trupmenų dalis naudojamas *metodas, pagristas dalinant verčiamą skaičių iš naujosios sistemos pagrindo.* Pagal (2) sveikas skaičius $A_{(q1)}$ sistemoje su pagrindu q_2 užrašomas

$$A_{(q2)}=b_{n-1}q^{n-1}{}_2+b_{n-2}q^{n-2}{}_2+\ldots+b_1q^1{}_2+b_0q^0{}_2$$

Perrašius šią išraišką pagal Honerio schemą, gausime

$$A_{(q2)}=(\ldots(((b_{n-1}q_2+b_{n-2})q_2+b_{n-3})q_2+\ldots+b_1)q_2+b_0) \qquad (9)$$

Skaičiaus užrašymas $A_{(q2)}$ pavidalu (9) leidžia išsiaiškinti nagrinėjamo metodo esmę. Jeigu dešinę (9) išraiškos pusę padalinti per pagrindą q_2, tai gausime sveikąją dalį $(\ldots(b_{n-1}q_2+b_{n-2})q_2+\ldots+b_1)$ ir liekaną b_0. Padaliję gautą dalį iš q_2, rasime antrąją liekaną b_1. Kartojant dalybos procesą n kartų, gausime paskutinę liekaną b_{n-1}. Verčiant visi aritmetiniai veiksmai turi būti atliekami pagal q_1 pagrindą turinčios skaičiavimo sistemos taisykles.

Paverčiant taisyklingas trupmenas ir skaičiavimo sistemų su pagrindu q_1 į sistemą su pagrindu q_2 naudojamas *metodas, bazuojantis dauginant verčiamą taisyklingą trupmeną iš naujos skaičiavimo sistemos pagrindo* q_2. Taisyklinga trupmena $A_{(q1)}$ skaičiavimo sistemoje, kurios pagrindas q_2 gali būti užrašyta

$$A_{(q1)}=b_{-1}q^{-1}{}_2+b_{-2}q^{-2}{}_2+\ldots+b_{-m}q^{-m}{}_2$$

Perrašius šią išraišką pagal Honerio schemą, gausime

$$A_{(q2)}=q^{-1}{}_2(b_{-1}+q^{-1}{}_2(b_{-2}+\ldots+q^{-1}{}_2(b_{-(m-1)}+q^{-1}{}_2b_{-m})\ldots)) \qquad (10)$$

Jeigu dešinę (10) išraiškos pusę padauginti iš q_2, tai surasime netaisyklingą trupmeną, kurios sveika dalis bus b_{-1}. Padauginus likusią trupmeninę dalį per pagrindą q_2, gausime trupmeną,

kurios sveikoje dalyje bus b_{-2}, ir t.t. Kartojant daugybos procesą m kartų, surasime visus skaičiaus trupmenines dalies m skaitmenis naujoje skaičiavimo sistemoje. Verčiant visi aritmetiniai veiksmai turi būti atliekami pagal q_1 pagrindą turinčios skaičiavimo sistemos taisykles.

Verčiant taisyklingas trupmenas iš vienos skaičiavimo sistemos į kitą, galima gauti begalinę arba diverguojančios eilutės pavidalo trupmeną. Skaičiaus vertimo procesą reikia užbaigti, kai bus trupmeninė dalis, turinti visuose eilėse nulius, arba bus pasiektas duotas vertimo tikslumas.

Verčiant netaisyklingas trupmenas iš vienos skaičiavimo sistemos į kitą, reikalingas atskiras vertimas sveikosios ir trupmeninės dalių, pagal aukščiau aprašytas taisykles. Gauti rezultatai užrašomi naujosios trupmenos pavidalu, skaičiavimo sistemoje su pagrindu q_2.

1.2. Bendros sąvokos

Operacinė sistema (OS) – organizuotas rinkinys programų, skirtų skaičiavimo resursams valdyti, skaičiavimo procesams planuoti ir organizuoti.

Duomenys – bet kokie informaciniai objektai skirti apdoroti ir saugoti ESM. Panašus duomenys grupuojami (pagal struktūrą, taikymo sritį ar kitus kriterijus) į aibes, vadinamas duomenų rinkiniais, ir joms suteikiami vardai.

Norėdamas apdoroti duomenis, ESM vartotojas turi pateikti *užduotį* ESM. Užduotį sudaro žingsniai; per vieną žingsnį vykdoma viena vartotojo arba sisteminė[1] apdorojimo programa. Užduoties žingsnio atlikimas – *procesas*, kurį pradeda operacinė sistema. Užduoties apraše nurodomi bendri visos užduoties duomenų rinkiniai ir kiti skaičiavimo resursai; be to, jie detalizuojami kiekvienam užduoties žingsniui. OS valdo būtent užduočių atlikimą.

Skaičiavimo resursai – tai aparatūrinės, programinės bei informacinės priemonės užduočiai atlikti.

Mašina iš karto gali spręsti keletą nepriklausomų užduočių, ir visoms joms reikia skaičiavimo resursų, kurie yra riboti. OS turi valdyti skaičiavimo resursus taip, kad vartotojų užduotys būtų atliekamos kuo efektyviau.

Atliekamos užduotys nuosekliai pereina parengties, atlikimo ir baigties fazes.

Parengties fazėje OS priskiria skaičiavimo resursus užduočiai atlikti. Tai atlieka OS programų valdomi sisteminiai procesai. Parengties fazė baigiasi sudarius proceso, kurį valdys pirmojo užduoties žingsnio programa, aprašą. Atlikimo fazėje sprendžiami užduoties žingsnių uždaviniai – vykdoma pagrindinė užduoties funkcija. Ši fazė baigiasi įvykdžius vartotojo programą. Baigties fazėje resursai grąžinami laisvų resursų fondui, išvedami rezultatai ir kt. Šios fazės, kaip ir parengties, uždavinius atlieka sisteminiai procesai.

Paprastai ESM tuo pačiu metu atlieka keletą užduočių, ir įvairios užduotys gali būti įvairiose fazėse, t.y. ESM gali būti sprendžiama kartu keletas sisteminių ir probleminių uždavinių. OS veikia, perjungdama centrinį apdorojimo įtaisą – centrinį procesorių (ar keletą jų) – pasirinktam procesui atlikti. Procesorius perjungiamas, kai pertraukiamas vykstantis procesas.

Pertraukimas – tai laikinas vykstančio proceso sustabdymas, norint pereiti prie kito proceso, kurį būtina arba tikslinga atlikti tuo metu.

1.3. Skaičiavimo resursai ir jų efektyvumas

Pagrindinė operacinės sistemos funkcija – valdyti skaičiavimo resursus. Skaičiavimo resursai – skaičiavimo mašinų mokslo, paprastai vadinamo skaičiavimo technika, objektas. Skaičiavimo resursai – tai visuma techninių, programinių ir informacinių priemonių, organizuojančių ir valdančių skaičiavimo procesą bei atliekančių pačius skaičiavimus. Tai:

- operaciniai resursai (procesoriai, informacijos mainų kanalai ir kiti panašus įtaisai);

[1] Programos skirstomos į taikomąsias (problemines) ir sistemines. Taikomąsias programas sudaro vartotojai savo konkretiems uždaviniams spręsti, o sisteminės programos yra bendrosios ESM programinės įrangos (žr. 2.b. sk.) dalis.

- atminties įtaisai;
- terminaliniai informacijos įvedimo ir išvedimo įtaisai (spausdinimo įtaisai, skeneriai, displėjai ir pan.);
- ryšio priemonės (ryšio linijos ir aparatūra informacijai tarp toli esančių įtaisų persiųsti);
- programinės įrangos priemonės (algoritminės kalbos, jų transliatoriai, operacinės sistemos, įvairios kitos sisteminės bei vartotojų programos, saugomos ESM informacinėse bibliotekose);
- informacinių duomenų bazės (organizuoti duomenų rinkiniai).

Taigi, pagal šią klasifikaciją, OS – taip pat skaičiavimo resursai, programinės įrangos priemonių aibės elementai. Iš esmės tai teisinga. Pagal aukščiau duotą apibrėžtį, OS – programų rinkinys; tuo tarpu šiuolaikiniu požiūriu OS sudaro tik valdymo programos. Tačiau greta gyvuoja ir kitas požiūris – OS traktuojamas kaip programinės įrangos priemonių visuma. Tokio požiūrio laikosi ir stambiausia duomenų apdorojimo pramonės sektoriaus firma IBM, jis „įteisintas“ ir vieningosios sistemos ESM programinės įrangos dokumentacijoje; čia OS sudaro praktiškai visa ESM programinė įranga – ir programos skaičiavimo resursams planuoti, skirstyti bei valdyti (valdymo programos), ir programos aukšto lygio programavimo kalbomis ar asembleriu parašytoms programoms transliuoti į mašinos kalbą (objektinį kodą) bei paruošti jas atlikti (apdorojimo programos), ir programos, skirtos specialiesiems tikslams (informacijos laikmenoms paruošti, duomenims iš vienų laikmenų į kitas perrašyti ir pan. – serviso programos), ir vartotojų parašytos probleminės programos (jos taip pat vadinamos apdorojimo programomis). OS apima tik bendros paskirties (bendrosios) programinės įrangos dalį – valdymo programas. Minėtosios programos (be vartotojo sudaromų probleminių) ir serviso programos – taip pat bendrosios programinės įrangos sudėtinės dalys. Bendrąsias apdorojimo ir vartotojo problemines programas OS valdo vienodai.

Kalbant apie operacines sistemas, reikia paminėti ir jos vartotojus, naudojančius ESM savo praktinėje veikloje. Tai – programuotojai ir aptarnaujantysis personalas. Programuotojų būrys gana gausus: pirmiausia tai profesionalūs ir neprofesionalūs programuotojai. Profesionalių programuotojų veiklos sritis – bendrosios programinės įrangos kūrimas, priežiūra ir taikymas konkrečiai ESM.

Dabartiniu metu, vis plačiau taikant ESM, daugėja ir neprofesionalių programuotojų. Šie programuotojai savo praktinius uždavinius sprendžia nesikreipdami į profesionalus, ankščiau buvusius tarpininkus tarp vartotojo ir ESM. Šių programuotojų ypač pagausėjo atsiradus asmeninėms ESM. Efektyvi ir „draugiška“ („draugiška“ ta prasme, kad operatyviai padeda vartotojui) programinė ESM įranga, visų pirma „draugiška“ operacinė sistema – sėkmingo šių programuotojų darbo ESM laidas.

Aptarnaujančiam personalui OS sudaro sąlygas lengvai kaupti duomenis apie tai, kaip naudojami skaičiavimo resursai, paprastai keisti ESM konfigūraciją sutrikus aparatūrai ir kitais atvejais, iš anksto paruošti informacijos laikmenų tomus su duomenų rinkiniais ir kt. Palaikydamos ryšį su ESM operatoriumi, OS sudaro sąlygas operatyviai valdyti užduočių atlikimą.

Operacinė sistema – efektyvaus skaičiavimo resursų naudojimo prielaida. Skaičiavimo resursų efektyvumas vertinamas kriterijumi

$$E = \frac{R}{C};$$

čia R – apibendrintas rezultatas, gautas panaudojus tam tikrus resursus, o C – apibendrintos išlaidos tam rezultatui gauti.

Rezultatas R priklauso nuo daug sunkiai aprašomų formulėmis komponentų. Tai, pavyzdžiui, skaitmeninė ir grafinė informacija, uždavinio spendimo laikas naudojantis tam tikrais skaičiavimo resursais, ESM patikimumas, naudojimosi patogumas ir kt. Išlaidos C apima išlaidas mašininiam laikui, programinei įrangai, programavimui ir pan. Svarbiausia programinė priemonė skaičiavimo resursų efektyvumui didinti – spartinti skaičiavimus sudarant lygiagrečiųjų skaičiavimų

procesus (tikrųjų lygiagrečiųjų skaičiavimu, kai keli ESM įtaisai atlieka operacijas tuo pačiu metu, ir virtualių lygiagrečiųjų skaičiavimų, kai tame pačiame laiko intervale dirba keli įtaisai paskirstymo laiko režimu), o tam reikia kurti specialią programinę įrangą. Taip pat svarbu, kad kuriama programinė įranga būtų efektyvi, nes, visų pirma, šios įrangos priemonės labai brangios, o antra vertus, jos linkusios visus kitus skaičiavimo resursus priversti sau tarnauti.

Iš viso to išplaukia, kad, didinant skaičiavimo resursų efektyvumą, OS efektyvumui tenka labai svarbus vaidmuo. Efektyvi OS gali padidinti skaičiavimo resursų efektyvumą, didindama ESM pralaidumą bei parengties koeficientą ir trumpindama atsako laiką.

ESM *pralaidumas* – atliekamo darbo apimtis per apibrėžtą laiką, pavyzdžiui, atliekamų užduočių skaičius per parą, pamainą ar pan. Jis priklauso tiek nuo aparatūros charakteristikų (atliekamų operacijų skaičiaus per sekundę ir kt.), tiek nuo programinės įrangos kokybės.

Atsako laikas – sistemos[2] (ESM) reakcijos į vartotojo užklausą laikas. Šis parametras gali priklausyti ne tik nuo OS, bet ir nuo vartotojų aptarnavimo režimo. Pavyzdžiui, kai ESM dirba paketų apdorojimo režimu, šis laikas beveik visuomet ilgesnis, nei kai ji dirba realaus ar paskirstyto laiko režimu.

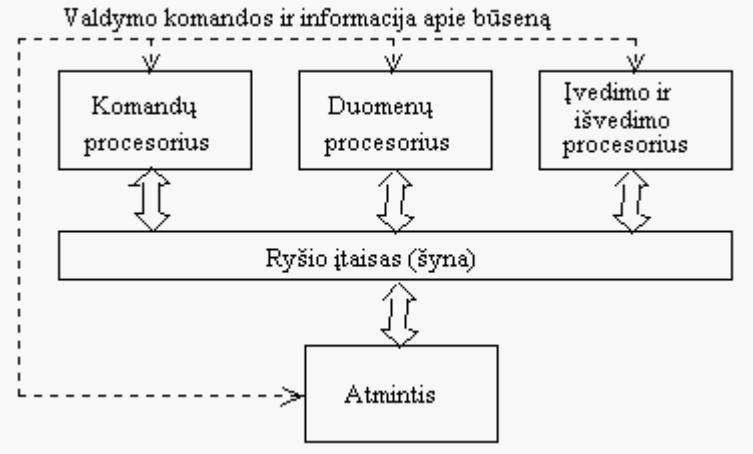
Parengties koeficientas atspindi sistemos pasirengimą atlikti vartotojų užduotis. Šis koeficientas mažas, kai dažnai genda ESM aparatūra ir modifikuojama programinė įranga. Jis padidėja prijungus papildomą aparatūrą ir įtraukus į OS klaidų aptikimo procedūras.

Be kita ko, koks svarbus OS vaidmuo didinant skaičiavimo resursų efektyvumą, matyti ir iš to, kaip didėja parengiamųjų darbų laiko ir sprendimo laiko santykis didėjant centrinio procesoriaus našumui. Parengiamieji darbai – tai operatoriaus atliekami veiksmai pereinant prie naujos užduoties: duomenų rinkinių montavimas įvedimo ir išvedimo ar išorinės atminties įtaisuose, įvairūs aparatūros perjungimai ir kt. Jei nuo vienos užduoties prie kitos pereinama neautomatiškai, ESM stovi, kol atliekami parengiamieji darbai. Automatiškai nuo vienos užduoties prie kitos pereinama pasinaudojus OS.

1.3. Von Neuman kompiuterio architektūra ir veikimo principai.

Dauguma šiandien dar naudojamų ir gaminamų kompiuterių sukurti remiantis pagrindiniais principais, kuriuos prieš metų suformulavo von Neuman, Burks ir Goldstine. Nors terminas "von Neumano kompiuteris" visiems seniai žinomas ir įprastas, tikslaus jo apibrėžimo nėra. Autoriai savo idėją suformulavo taip:
Principinės mašinos komponentės yra:
1) kadangi tai turi būti bendros paskirties skaičiavimo mašina, joje turi būti tokie įtaisai, kaip aritmetinis, atminties, valdymo ir ryšio su žmogumi; mašina bus pilnai automatinė;
2) mašina turi saugoti atmintyje tokia pačia forma ne tik skaičiavimams reikalingą skaitmeninę informaciją (duomenis, funkcijų lenteles, tarpinius rezultatus), bet ir komandas, valdančias skaičiavimą; skaičiavimo mašinoje turi būti įtaisas programoms saugoti, o taip pat įtaisas, kuris jas suprastų ir valdytų jų vykdymą;
3) konceptualiai mes apsvarstėme dvi skirtingas atminties formas - skaičių ir komandų atmintį; jei komandos bus koduojamos skaičiais ir jei mašina sugebės atskirti jas nuo skaičių, viskas gali būti saugoma vienoje atmintyje;
4) jei atmintis tik saugo komandas, mašinoje turi būti organas, kuris automatiškai jas vykdo. Šį organą vadinsime valdymu;
5) kadangi tai yra skaičiavimo mašina, joje turi būti aritmetinis įtaisas, kuris vykdytų aritmetines operacijas (sudėtį, atimtį, daugybą, dalybą), o taip pat ir kitas dažnai sutinkamas operacijas;
6) joje taip pat turi būti įvedimo ir išvedimo įtaisas, kuris leistų operatoriui komunikuoti su mašina.
Pagal šį paaiškinimą galima nubraižyti tokią kompiuterio struktūrą:

[2] Sistema bus vadinama skaičiavimo resursų visuma.



Šiame paveiksle paprastumo dėlei priimta, kad visi informacijos persiuntimai tarp kompiuterio įtaisų vykdomi per vienintelę šyną. Komandų ir duomenų procesorius įprasta apjungti į vieną įtaisą - **centrinį procesorių** (**CPU**).

Iš kompiuterių organizacijos taško Neumano architektūrą galima apibūdinti kaip ***minimalios aparatūros architektūrą***, kurią sudaro tokie įtaisai:
1) **CPU**, apjungiantis **operacinį** (duomenų procesorių) ir **valdymo** (komandų procesorių) **įtaisus**; CPU interpretuoja ir vykdo programos komandas (išskyrus įvedimo ir išvedimo komandas);
2) **atmintis**, kurioje saugomi visi duomenys ir programos;
3) **įvedimo ir išvedimo įtaisas**, kuris kartu su periferiniais įtaisais užtikrina kompiuterio ryšį su aplinka;
4) vidiniai duomenų keliai - **magistralės**, kurios užtikrina informacijos mainus tarp visų kompiuterio įtaisų.

Von Neumano architektūros funkcionavimo principo esmė - operacija atliekama visuomet su turiniu ląstelės, kuri nurodyta komandoje, neatsižvelgiant į tai, ką šios ląstelės turinys reiškia. Šį principą galima pavadinti minimalios atminties panaudojimo principu. Mažiausias identifikuojamas informacijos vienetas kompiuteryje - dvejetainis vektorius, kurio ilgį pažymėsime L. Šis vektorius von Neumano kompiuteryje atitinka tokiu informacijos tipus:

- **komandas**,
- **duomenis** (skaičius, dvejetainius vektorius ar simbolius),
- atminties ląstelių arba įvedimo ir išvedimo įtaisų **adresus**.
  Šiuolaikiniuose kompiuteriuose (ne von Neumano tipo) galima sutikti ir kitokius informacijos tipus:
- **tegus** (*tags*) - bitų grupes, kurios nurodo palydimos informacijos tipą;
- informacijos vienetų **deskriptorius**;
- informacijos vienetų **identifikatorius** (vardus).

Von Neumano kompiuteryje atminties lastelės turinys savęs neidentifikuoja. Jis interpretuojamas pagal tokią schemą:
a) pirmiausia kompiuteris pagal savo būseną nustato, kuriame interpretacijos žingsnyje jis yra, t.y., ar šis žodis turi būti interpretuojamas kaip komanda, ar kaip duomenys;
b) duomenų interpretaciją apsprendžia vykdomos komandos tipas (operacijos kodas); šios komandos argumentu ir yra duomenys.

Von Neumano kompiuterio funkcionavimo principai sąlygoja kietą dviejų fazių schemą programos vykdyme: pirmojoje fazėje išrenkama ląstelė, kurios turinys bus interpretuojamas kaip komanda. Antrojoje fazėje ši komanda vykdoma, pagal komandoje esančią informaciją iš atminties išrenkant ląstelę, kurios turinys interpretuojamas kaip apdorojamas operandas.

Detaliau nagrinėjant komandos vykdymo procesą, galima išskirti daugiau smulkesnių žingsnių, pavyzdžiui:
1. Išrinkti iš atminties komandą ir įrašyti ją į komandos registrą.
2. Pakeisti komandos skaitiklio, kuris nurodo vykdomos komandos adresą, turinį.
3. Nustatyti išrinktos komandos tipą.
4. Jei komandoje nurodyti duomenys yra atmintyje, nustatyti jų vietą.

5. Jei reikia, išrinkti iš atminties duomenis ir perduoti juos į CPU registrus.
6. Vykdyti komandoje nurodytą operaciją.
7. Rezultatus įrašyti į nurodytą vietą.
8. Pereiti į 1 žingsnį kitai komandai vykdyti.

Tokio paprasto kompiuterio interpretatorius Paskalyje gali būti aprašytas tokia procedūra:

```
type word    = ...;
     address = ...;
     mem     = array [0..    ] of word;

Procedure interpreter (var memory : mem; var ac : word; start     : address);
var ProgramCounter, Datalocation : address;
    InstrRegister, data          : word;
    DataNeeded                   : boolean;
    RunBit                       : 0..1;
begin
ProgramCounter := start;
RunBit := 1;

while RunBit=1 do begin
{ išrinkti komandą }
 InstrRegister := memory[ProgramCounter];

{ pakeisti programų skaitiklio turinį }
ProgramCounter := ProgramCounter + 1;

{dekoduoti komandą }
DetermineInstrType (  InstrRegister , InstrType  );

{ nustatyti duomenų vietą }
FindData ( InstrRegister , InstrType , DataLocation, DataNeeded );

{ išrinkti duomenis  (jei reikia) }
if  DataNeeded  then data := memory [ DataLocation ]:

{ vykdyti operaciją }
```

```
Execute ( InstrType , data, memory, ac, ProgramCounter, RunBit );
end
end;
```

Interpretatoriumi aprašomos mašinos detalesnė struktūra pateikta paveiksle. Čia pažymėta:

- **T** - laikinojo saugojimo (papildomas) registras,
- **IR** - komandos registras,
- **PC** - programos skaitiklis,
- **M** - atmintis,
- **AR** - jos adreso registras,
- **DR** - jos duomenų registras.

2. Intel mikroprocesorių architektūros ypatybės

2.1. ESM Struktūra

2.1.1. Centrinis procesorius

Registrai. Registrai – vietinės centrinio procesoriaus atminties įtaisai, kuriose fiksuojama ESM operatyvinio valdymo informacija. Ši informacija apima: vykdomosios komandos adresą, įvykdytos komandos rezultato požymius, operacinės CP būsenos požymius (nustatančius CP vykdomų komandų aibę), leidžiamų pertraukimų požymius ir kitus komponentus. Dalis šios informacijos gali būti fiksuojama bendros paskirties (bendruosiuose) registruose, dalis – specialiuose būsenos registruose; įvairiose ESM yra savas registrų paskirstymas ir savi pavadinimai. Būsenos registruose fiksuojami tie būsenos komponentai, kuriuos išsaugo aparatūra pertraukimo metu (įrašydama būsenos registrų turinį į pagrindinę atmintį), t.y. juose atsispindi vykstančio proceso būsena.

Savo programose vartotojas gali kreiptis ne į visus CP registrus (suprantama, jei jis programuoja asemblerio kalba). Vartotojui prieinami bendrieji registrai ir kai kurie būsenos registrų komponentai, pavyzdžiui, įvykdytos komandos rezultato požymių indikatoriai. Tuo tikslu ribojamas vartotojui leidžiamų komandų rinkinys. Kitaip vartotojas galėtų neleistinai pakeisti ESM būseną, pakenkdamas sau ir kitiems, kurių užduotys atliekamos tuo pačiu metu. Keisti ESM (procesų) būsenas tam tikslui skirtomis komandomis gali tik privilegijuotas ESM „vartotojas“ – operacinė sistema.

8086/88 mikroprocesorius turi daug dviejų baitų **vidinių registrų,** kiekvienam iš kurių priskirtas „vardas“, arba mnemonika. Visus registrus sąlygiškai galima suskirstyti į 3 grupes:

1) bendros paskirties registrai – AX, BX, CX, DX;
2) registrai – rodyklės – DI, SI, BP, SP, IP;
3) segmentiniai registrai – CS, DS, ES, SS.

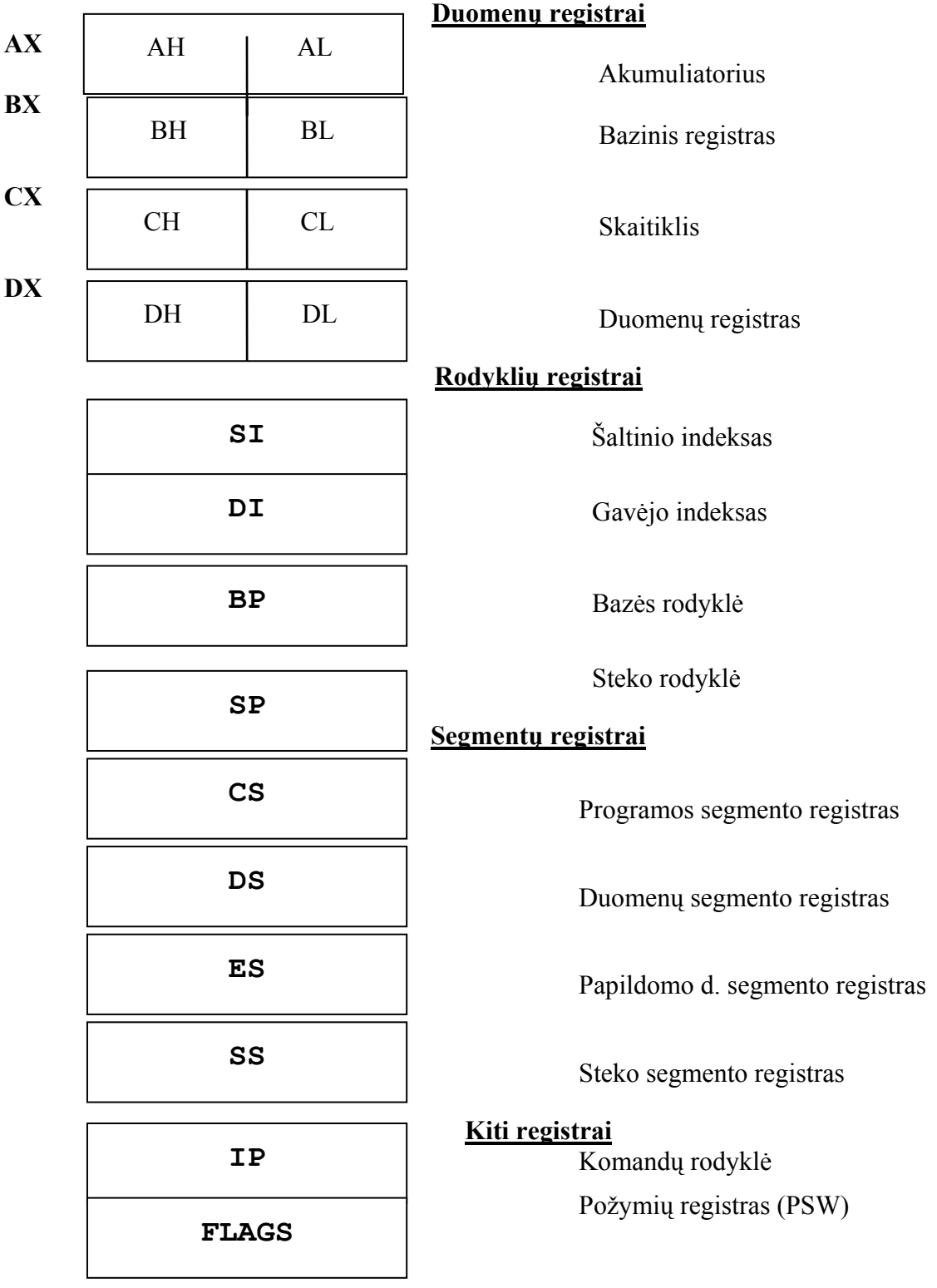
Bendrosios paskirties registrai leidžia kreiptis prie ištiso žodžio registre (mnemonikos AX, BX, CX, DX) arba atskirų baitų (mnemonikos AH, BH, CH, DH) – prie vyresniojo baito, o AL, BL, CL, DL – prie jauniausio baito). Jie naudojami tarpiniam operandų saugojimui. Kadangi registrai pasiekiami daug kartų greičiau negu operatyviosios atminties ląstelės ir tarpiniai rezultatai patalpinami registruose, tame tarpe ir bendrosios paskirties registruose, laimimas programos vykdymo greitis.

Registrai – rodyklės leidžia pasiekti tik ištisą žodį registre. IP (Instruction Pointer) registras poroje su CS registru (CS:IP) nusako atmintyje kitos mašininės komandos fizinį adresą. IP reikšmė keičiama tik sąlyginių ir besąlyginių perėjimų komandomis, ciklu, kreipiantis į procedūras ir vykdant pertraukimus. SI ir DI registrus naudoja mikroprocesoriaus aparatūra kai kuriose mašininėse komandose. Tai vadinamieji **grandininiai primityvai.** Vykdant tokias komandas (baitų grupes arba žodžių persiuntimas, baitų grupės arba žodžių palyginimas ir pan.) šaltinio fizinis adresas nusakomas DS:SI reikšme, o fizinis tikslo adresas – ES:DI. Po kito baito apdorojimo procesorius automatiškai modifikuoja registrų DI ir SI reikšmes. Iš čia kilę ir registrų vardai: DI (Destination Index – tikslo indeksas), SI (Source Index – šaltinio indeksas). Kadangi šios komandos pakankamai retos, daugelis kompiliatorių naudoja DI ir SI registro kintamiesiems saugoti.

Registras SP poroje su SS nusako esamos **steko viršūnės** fizinį adresą. Įrašymas į steką vyksta arba specialiosiomis PUSH instrukcijomis, arba mikroprocesorius įrašo ten gražinimo taško adresą, vykdant procedūrų iškvietimo instrukcijas (esamos reikšmes CS ir IP FAR – perėjimams arba tik IP NEAR - perėjimams) ir pertraukimus. Kiekvieną kartą įrašant žodį į steką SP reikšmė aparatūriškai sumažėja du kartus, o kiekvieną kartą nuskaitant žodį iš steko – padidėja du kartus. Todėl sakoma, kad stekas „auga“ iš viršaus į apačią (t.y. nuo didesnių adresų link mažesnių, o steko kaip buferio darbo mechanizmą vadina LIFO (Last In, First Out)). BP registras yra rodyklė, susieta

pagal nutylėjimą su steko segmentu. Jeigu mašininėje komandoje operandas nusakomas nuoroda į BP, tai fizinis operando adresas formuojamas atmintyje panaudojant SS.

Procesoriaus registrai



2.2. Intel šeimos istorija

- **Intel 4004, pirmasis mikroprocesorius (1971 lapkr.).**

Tai pirmasis vienkristalis 4 bitų mikroprocesorius, skirtas kalkuliatoriui. Duomenų ilgis - 4 bitai, komandų - 8 bitai. Duomenims ir programoms saugoti naudojamos atskiros atmintys: 1 KB - duomenims, 4 KB - programoms. PC ilgis - 12 bitų. Turėjo 16 registrų (po 4 bitus), 4 lygių steką kreipiniams į paprogrames.
Komandų skaičius - 46; dažnis - 108 KHz, 2300 tranzistorių (10 mikronų).

- **Intel 8080 (1974)**

Intel 8080 mikroprocesorius turėjo 16 bitų adreso bus ir 8 bitų duomenų magistrales. Jo viduje - septyni 8 bitų registrai (A-E, H ir L galėjo būti naudojami poromis - BC, DE ir HL - apjungus į 16 bitų registrus), 16 bitų steko rodyklę ir 16 bitų programos skaitiklį. Adresuojama atmintis - 64 KB.
Tranzistorių skaičius - 6 000 (6 mikronai), dažnis - 2 MHz.
Po poros metų Intel išleido 8085, turintį dvi naujas komandas trims papildomiems pertraukčių kanalams valdyti. Maitinimas - +5V, taktų generatorius ir magistralės valdymo schemos įdėtos į vidų.
Tranzistorių skaičius - 6 500 (3 mikronai), dažnis - 5 MHz.

- **Intel 8086 (1978)**

Intel 8086 skirtas 8080/8085 sistemoms modernizuoti: programiškai suderinamas su 8080, turintis panašų registrų rinkinį, tačiau duomenų ilgis išplėstas iki 16 bitų. Magistralės Interfeiso Įtaisas aprūpino vykdymo įtaisą komandomis per 6 baitų išankstinio išrinkimo eilę, todėl komandų išrinkimas vykdomas kartu su kitų komandų vykdymu. Adresuojama atmintis - 1 MB.
Keturi 16 bitų bendrieji registrai gali būti naudojami kaip 8 atskiri 8 bitų registrai. Be to, 8086 turi keturis 16 bitų registrus (įskaitant steko rodyklę ir keturis 16 bitų segmentų registrus.
Tranzistorių skaičius - 29 000 (3 mikronai), dažnis - 4,77 MHZ., kaina - 360 $.
1979 metais Intel išleido 8088 - 8086 variantą, turintį 8 bitų duomenų magistralę.
1980 metais Intel išleido 8087 - slankaus taško (matematinį) koprocesorių.

- **Intel 80286 (1984)**

1984 metais Intel išleido 80286 - 8086 pakeisti skirtą mikroprocesorių, plačiai naudotą IBM PC/AT sistemose. Adresuojama atmintis - 16 MB, virtuali atmintis - 1 GB.
Tranzistorių skaičius - 134 000 (1,5 mikronų), dažnis - 6 MHz.
1985 metais Intel išleido 80287 - slankaus taško (matematinį) koprocesorių.

- **Intel 80386 (1985)**

Intel 80386 išplėstos adresavimo galimybės, pridedant indekso daugiklį (bazės reg + indekso reg * daugiklis (1, 2, 4 ar 8) + poslinkis (8 arba 32 bitų konstanta = 32 bitų adresas). Pridėtas atminties dispečeris (mikronų), apsaugos režimai (vadinamieji žiedai). Adresuojama atmintis - 4 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 275 000 (1,5 mikrono), dažnis - 16 MHz., kaina - 299 $.
1987 metais Intel išleido 80386SX - su 16 bitų duomenų magistrale. Adresuojama atmintis - 16 MB, virtuali atmintis - 256 GB.
Tranzistorių skaičius - 275 000 (1,5 mikrono), dažnis - 16 MHz.
1990 metais Intel išleido 80386SL - pirmąjį mikroprocesorių portatyviems kompiuteriams. Adresuojama atmintis - 4 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 855 000 (1 mikronas), dažnis - 20 MHz.
1987 metais Intel išleido 80387 - slankaus taško (matematinį) koprocesorių.

- **Intel 80486 (1989)**

Įvestas komandos vykdymo konvejeris, bendras vidinis 8KB kešas, integruotas FPU.
Adresuojama atmintis - 4 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 1,2 mln. (1 mikronas; 50 MHZ - 0,8 mikrono), dažnis - 25 MHZ., kaina - 900 $.
Be pagrindinio varianto (DX) buvo leidžiami 80486DX2 (su dažnio dvigubinimu viduje) ir 80486DX4 (su dažnio trigubinimu).
1991 metais Intel išleido 80486SX - be FPU. Adresuojama atmintis - 16 MB, virtuali atmintis - 256 GB. Tranzistorių skaičius - 1 185 000 (1 mikronų; 0,9 mln. - 0,8 mikronų), dažnis - 16 MHz.
1992 metais Intel išleido 80486SL - portatyviems kompiuteriams. Adresuojama atmintis - 64 MB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 1,4 mln. (0,8 mikrono), dažnis - 20 MHz.

- **Pentium (1993)**

Tai superskaliarinis (iki dviejų komandų sveikųjų skaičių įtaisuose ir vienos FPU) procesorius, turintis atskirus 8KB komandų ir duomenų kešus. Išorinė duomenų magistralė - 64 bitų.
Adresuojama atmintis - 4 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 3,1mln. (0,8 mikrono), dažnis - 60 MHZ., kaina - 878 $.

- **Pentium Pro (1995)**

Dar vadinamas "P6". Jį sudaro 1 ar 2-kristalai (CPU plius 256KB arba 512KB L2 kešas). CPU viduje turi atskirus 8KB komandų ir duomenų kešus. Procesorius superkonvejerizuotas: jo pakopų skaičius - 14. Konvejerio efektyvumui padidinti naudojama šakojimosi prognozė *(multiple branch prediction*) ir registrų pakeitimas (*register renaming*). Trys dekodavimo įtaisai (vienas - sudėtingoms komandoms, du kiti - paprastoms) dekoduoja vieną 80x86 komandą į mikrooperacijas. Vienu metu gali būti vykdomos penkios (dažniausiai - trys) mikrooperacijos, Tam skirti šeši funkciniai - FPU, 2 sveikųjų skaičių, 2 adresavimo, 1 įkrovimo/įrašymo.
Adresuojama atmintis - 64 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 5,5 mln. (0,35 mikrono), 256KB L2 kešas - dar 15,5 mln. tranzistorių, o 512KB - 31 mln.; dažnis - 150 MHZ., kaina - 974 $.

- **Pentium MMX (1997)**

Komandų sistema papildyta 57 naujomis komandomis, skirtomis grafiniams duomenims apdoroti. Tam naudojamas SIMD principas.
Adresuojama atmintis - 4 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 4,5 mln. (0,35 mikrono), dažnis - 166 MHz.
1997 m. buvo išleistas Mobile Pentium MMX - portatyviems kompiuteriams Adresuojama atmintis - apie 68 GB.
Tranzistorių skaičius - 4,5 mln. (0,25 mikrono), dažnis - 200 MHz.

- **Pentium II (1997)**

Skirtas brangiems personaliniams kompiuteriams. Turi 512 KB L2 kešą, prijungtą per atskirą magistralę (FSB - Front Side Bus). Sistemos magistralė - iki 100 MHz.
Adresuojama atmintis - 64 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 7,5 mln. (0,25 mikrono), dažnis - 266 MHz.

1998 m. buvo išleistas Mobile Pentium II - portatyviems kompiuteriams Adresuojama atmintis - apie 68 GB. Branduolio maitinimo įtampa - 1,7V.
Tranzistorių skaičius - 7,5 mln. (0,25 mikrono), dažnis - 233 MHz.

- **Celeron (1998)**

Skirtas nebrangiems personaliniams kompiuteriams. Neturi L2 kešo.
Adresuojama atmintis - 4 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 7,5 mln. (0,35 mikronų), dažnis - 266 MHz. Didesnio dažnio procesorių tranzistorių skaičius - 19 mln. (0,25 mikrono).

- **Pentium II Xeon (1998)**

Skirtas serveriams. Turi 512 KB, 1MB arba 2MB L2 kešą, prijungtą per atskirą magistralę (FSB - Front Side Bus). Sistemos magistralė - 100 MHz.
Adresuojama atmintis - 64 GB, virtuali atmintis - 64 TB.
Tranzistorių skaičius - 7,5 mln. (0,25 mikrono), dažnis - 400 MHz.

Intel procesoriai

Karta	Tipas	Data	D/A magistr. plotis	Vidinis kešas	Dažnis (MHz)	Mag. dažnis (MHz)
I	8088	1979	8/20	Nėra	4,77-8	4,77-8
I	8086	1978	16/20	Nėra	4,77-8	4,77-8
II	80286	1982	16/24	Nėra	6-20	6-20
III	80386DX	1985	32/32	Nėra	16-33	16-33
III	80386SX	1988	16/32	Nėra	16-33	16-33
IV	80486DX	1989	32/32	8 KB	25-50	25-50
IV	80486SX	1989	32/32	Nėra	25-50	25-50
IV	80486DX2	1992	32/32	8 KB	50-80	25-40
IV	80486DX4	1994	32/32	8 KB+8 KB	75-120	25-40
V	Pentium	1993	64/32	8 KB + 8 KB	60-200	60-66
V	Pentium MMX	1997	64/32	16 KB + 16 KB	166-233	66
VI	Pentium Pro	1995	64/32	8 KB + 8 KB	150-200	66
VI	Pentium II	1997	64/32	16 KB + 16 KB	233-450	66 - 75

3. Operacinių sistemų raida

3.1. Kompiuterių istorijos faktai ir skaičiai

Žmogaus noras spartinti skaičiavimus pasireiškė paprasčiausių skaičiavimo įtaisų - abako, skaitytuvų - kūrimu, vėliau pasirodė mechaniniai skaičiavimo įtaisai - aritmometrai. **Abakas** buvo sukurtas dar prieš mūsų eros pradžią ir plačiai paplito Japonijoje ir Kinijoje. Daug šimtmečių abakas ir jo modifikacijos liko vieninteliu skaičiavimo įtaisu. Įgudęs skaičiuotojas galėtų sėkmingai varžytis su mechaniniais kalkuliatoriais.
1642 m. prancūzų mokslininkas B.Paskalis sukonstravo pirmąjį **mechaninį skaičiavimo įtaisą**, kuris galėjo paskaičiuoti skaičių sumą ir skirtumą.
Jį patobulino vokiečių mokslininkas G.Leibnicas, 1673 metais sukūręs **mechaninę skaičiavimo mašiną**, galėjusią vykdyti visus keturis aritmetinius veiksmus. B.Paskalio ir G.Leibnico įtaisai buvo netobuli, nes technikos ir technologijos išvystymo lygis XVII amžiuje buvo žemas, tačiau jie laikomi skaičiavimo technikos istorijos pradžia.
XIX amžiuje pažymėtini dviejų mokslininkų darbai, turėję didelę įtaką skaičiavimo technikos vystymui:

- 1822 m. anglų matematikas Č.Bebidžas sukūrė **"skirtuminę" mašiną**, kuri galėjo paskaičiuoti antrojo laipsnio polinomų reikšmes gana dideliu tikslumu (iki 8 ženklų); 1833 m. jis norėjo sukonstruoti programa valdomą mechaninį skaičiavimo įtaisą, tačiau iki galo realizuoti savo sumanymą nepajėgė.
- 1854 m. kitas anglų matematikas Dž.Būlis išleido **knygą "Mąstymo dėsniai"**, kurioje išdėstė teiginių logiką, vėliau gavusią **Būlio algebros** vardą.

Praėjus daugiau nei 100 metų Č.Bebidžo pasiūlytą programinio valdymo idėją realizavo **Dž.Atanasovas** penktojo dešimtmečio pradžioje JAV (jo kompiuteris ABC buvo skirtas fizikos lygtims spręsti, todėl jis artimesnis kalkuliatoriui, nei kompiuteriui), **K.Cūzė** 1942 m. Vokietijoje (dėl antrojo pasaulinio karo šis kompiuteris ilgai nebuvo žinomas) ir **H.Aikenas** 1944 m. JAV (tai buvo skaičiavimo įtaisus iš elektromagnetinių relių, o valdymo programą užkoduojama perfojuostoje).
Dž.P.Eckertas ir Dž.Močli (Pensilvanijos universitetas) JAV kariškių pavedimu dar karo metais sukūrė pirmąjį elektroninį bendrosios paskirties kompiuterį **ENIAC** (Electronic Numerical Integrator and Calculator), informacija apie kurį buvo paskelbta tik 1946 m. Šis kompiuteris buvo skirtas artilerijos šaudymo lentelėms skaičiuoti. U formos kompiuteris buvo 24 metrų ilgio, 2,5 m. aukščio ir svėrė apie 30 tonų. Viso kompiuteryje buvo panaudota 18000 elektroninių lempų. Tai buvo programuojamas kompiuteris. Tiesa, programavimas buvo atliekamas rankiniu būdu, atitinkamai sujungiant kabelius ir nustatant perjungėjus. Duomenys buvo įvedami naudojant perfokortas. Skaičiavimams reikalingų programų "įvedimas" trukdavo net visą dieną.
1944 m. Dž. fon Neimanas buvo pakviestas dirbti prie ENIAC projekto. Siekdami supaprastinti programavimą, projekto autoriai parengė **atmintyje saugomos programos panaudojimo idėją**. Šis pasiūlymas buvo paskelbtas ir davė pradžią plačiai žinomai sąvokai "Fon Neimano kompiuteris". Pirmuoju kompiuteriu, kuriame programa saugojama jo atmintyje, laikomas 1949 m. M. Uilkso realizuotas **EDSAC** (Electronic Delay Storage Automatic Calculator) projektas Kembridže. Pirmuoju komerciniu kompiuteriu laikomas **UNIVAC I** (1951 m.). Jis kainavo apie 1 mln. dolerių. Tokių kompiuterių buvo pagaminta 48.
IBM firma, kuri buvo žinoma kaip perfokortų ir įstaigų automatizavimo įrangos gamintoja, 1950 m. įsijungė į kompiuterių gamybą. Jos pirmasis kompiuteris buvo IBM 701, kurių pagaminta 19. Tais laikais vyravo nuomonė, kad kompiuteris - labai specializuotas įtaisas, turintis siaurą rinką. Šią nuomonę sulaužė IBM, investavusi 5 mlrd. dolerių ir 1964 m. paskelbusi apie sistemą **IBM/360**,

kurią tuo metu sudarė 6 modeliai, besiskiriantys savo kaina ir našumu iki 25 kartų. 1965 m. DEC pristatė PDP-8 - pirmąjį komercinį **minikompiuterį** - mažą ir palyginus pigią (apie 20 000 dol.) mašiną. Minikompiuteris laikomas šiuolaikinių mikroprocesorių protėviu. 1963 m. buvo pristatytas pirmasis **superkompiuteris** - CDC 6600. Jo autorius S.Krėjus vėliau paliko CDC ir įkūrė savo firmą (Cray Research), kurioje 1976 m. sukūrė Cray-1, kuris tuo metu buvo greičiausias ir brangiausias pasaulyje, tačiau moksliniams uždaviniams turėjo geriausią našumo ir kainos santykį.
Po metų, 1977 m., S.Džobs ir S.Vozniakas davė pradžią **personalinių kompiuterių** pramonei, sukurdami Apple II. Tačiau po 4 metų IBM perėmė personalinių kompiuterių gamybos vairą į savo rankas. Dabar IBM ar IBM tipo personaliniai kompiuteriai tvirtai vyrauja rinkoje.

Pagrindinių paminėtųjų įvairių laikų kompiuterių duomenys pateikti šioje lentelėje:

Metai	**Modelis**	**Dydis (dm^3)**	**Galia (W)**	**Našumas (op/s)**	**Atmintis (MB)**	**Kaina ($)**	**Našumas/kaina**
1951	UNIVAC	28000	124500	1900	48	1000000	1
1964	IBM S360/mod. 50	1680	10000	500000	64	1000000	263
1965	PDP-8	225	500	330000	4	13000	10855
1976	Cray-1	1625	60000	166000000	32768	4000000	21842
1981	IBM PC	30	150	240000	256	3000	42105
1991	HP 9000/mod. 750	56	500	50000000	16384	7400	3556188

Kaip matome, kompiuterių našumo santykis su kaina per 40 metų pasikeitė daugiau nei 3,5 mln. kartų, o jei įvertinsime infliaciją, tai šis santykis pasikeitė daugiau nei 16 mln. kartų!

3.2. Kompiuterių kartos

Karta	**Technologija ir architektūra**	**Programinė įranga ir taikomosios programos**	**Atstovai**
Pirmoji (1945-54)	Elektroninės lempos, relinės atmintys, akumuliatorius ir PC valdomas CPU, fiksuoto taško aritmetika	Mašinos kalba/asembleris, vienas vartotojas, paprastos paprogramės, programa valdomi duomenų mainai	ENIAC, IBM 701
Antroji (1955-64)	Tranzistoriai, magnetinių šerdžių atmintys, slankaus taško aritmetika, I/O procesoriai, multipleksuoti kreipiniai į atmintį	Aukšto lygio progr. kalbos ir kompiliatoriai, paprogramių bibliotekos, komandiniai failai	IBM 7090, CDC 1604, Univac LARC
Trečioji (1965-74)	Mažo ir vidutinio integracijos lygio IS, mikroprogramavimas, konvejerizacija, kešai, lookahead procesoriai	Multiprogramines ir laiko paskirstymo OS, daugiavartotojiškos sistemos	IBM 360/370, CDC 6600, TI-ASC, PDP-8
Ketvirtoji (1975-90)	Didelio/labai didelio integracijos lygio IS, puslaidininkinės atmintys, multiprocesoriai, vektoriniai superkompiuteriai,	Multiprocesorių OS, kalbos, kompiliatoriai ir aplinkos lygiagrečiam apdorojimui	VAX 9000, Cray-X-MP, IBM 3090

	multikompiuteriai		
Penktoji (1991-)	ULSI/VHSIC procesoriai, atmintis ir komptatoriai, didelis pakavimo tankis, skaliarinės architektūros	Masyvus lygiagretus apdorojimas, heterogeninis apdorojimas	Fujitsu VPP500, Cray/MPP, Intel Paragon

Analizuodami, kaip per šį laikotarpį vystėsi kompiuterių programinė įranga, tyrinėtojai taip pat išskiria kompiuterių kartas. Ši periodizacija nesutampa su anksčiau pateiktu kompiuterių kartų sąrašu.

Karta	Pradžia	Būdingi programinės įrangos kartų požymiai
0	1947	Įkrovėjai, įvedimo-išvedimo paprogramės
1	1959	Magnetinių juostų įvedimo-išvedimo valdymo sistema; aukštesnių programavimo kalbų kompiliatoriai
2	1963	Diskinė operacinė sistema; multiprogramavimas
3	1970	Universali laiko skirstymo operacinė sistema; duomenų bankų sistemos
4	1978	Programinės įrangos perkėlimas į mikroprogramas (firmware); globaliniai ir lokaliniai tinklai; asemblerio pakeitimas sistemine programavimo kalba; programinės įrangos kūrimo įrankiai
5	1990	Žinių bazių sistemos; natūralios kalbos interfeisas; objektams orientuotos funkcinės ir loginės kalbos

M.Flynas, remdamasis vartotojui reikalingų funkcijų realizacijos technologijomis, apibrėžė tokius kompiuterių laikmečius:

- ankstyvasis laikmetis - vienprogramis darbas;
- vidurinysis laikmetis - daugiaprogramis darbas (paketinis apdorojimas);
- vėlyvasis laikmetis - laiko paskirstymas, globaliniai tinklai;
- paskirstytų sistemų laikmetis;
- dirbtinio intelekto laikmetis.

1950-ieji ***IOCS*** - įvedimo-išvedimo valdymo sistemos

Asembleriai ir A-IOCS jungtinės sistemos

Aukšto lygio programavimo kalbos: FORTRAN, ALGOL
Programos vykdymo etapai:

asemblerio pakrovimas ⟶ asembl. programos skaitymas ⟶ programos lokalizavimas ir vykdymas ⟶ duomenų įvedimas

1960-ieji **Pertraukimai, virtualinė atmintis, *JCL***

Pirmosios OS: Atlas, IBM System 360 (DOS)... **Naujos savybės:**

Paketinis režimas (*batch processing*), Multiprogramavimas, darbų skirstymas, laiko skirstymas, pagrindinis ir foninis režimai (*fore- & background, spooling*).

Realaus laiko sistemos, kompiuterinis valdymas

1970-1990 **Daugiaprocesorinės sistemos, lygiagretusis programavimas, Paskirstytos sistemos, tinklo sistemos, protokolai.**

Dialoginės sistemos, mini- (darbo stočių) ir personalinių kompiuterių OS, Grafinis vartotojo interfeisas, Hierarchinė informacijos struktūra, failų sistemos, kliento-serverio architektūra.

3.3. Šiuolaikinių operacinių sistemų istorija

MS-DOS	Windows	UNIX
		Bell Labs, projektas Multics vs UNIX, Ken Thomson ~1969
CP/M (Digital Research)		UNIX perrašytas nauja C kalba, Dennis Ritchie, 1973
CP/M, 86-DOS		1979 UNIX TSS v.7 Naujas Shell ir C; 1980 IBM - XENIX
MS-DOS 1.0, IBM PC	Apple OS	1981
MS-DOS 2.0, PC/XT instaliuojamos tvarkyklės	Apple OS	1983 System III,SCCS
MS-DOS 3.x, PC/AT 20M, RAM diskai, failų atributai, 3'' diskai, tinklo palaikymas.	Windows v.1.0; 1985	1985 AT&T System V, (SVR4, SVID)
MS-DOS 4.0; iki 2GB, komandų aplinka (Shell)	Windows v. 2.0; 1988	1988; X-Windows v.11 Kliento-serverio sistema
MS-DOS 5.0; efektyvus RAM valdymas, naujos komandos ir Shell; 1991	Windows v. 3.0; 1990 Windows v. 3.1; 1991 386 protected mode iki 16M RAM, OLE, TTF	~ Solaris (OS/2 v.1.0)
	WfW v. 3.11 1992 Windows NT 3.1, neprikl. nuo MS-DOS	
MS-DOS 6.x; 1994 disko kompresija, atminties planavimas	Windows NT v. 3.5 32 bitų	~ LINUX
MS-DOS 7.0; 1995	Windows'95	
MS-DOS ?.0; 1998	Windows 98	

4. Operacinių sistemų sąvokos

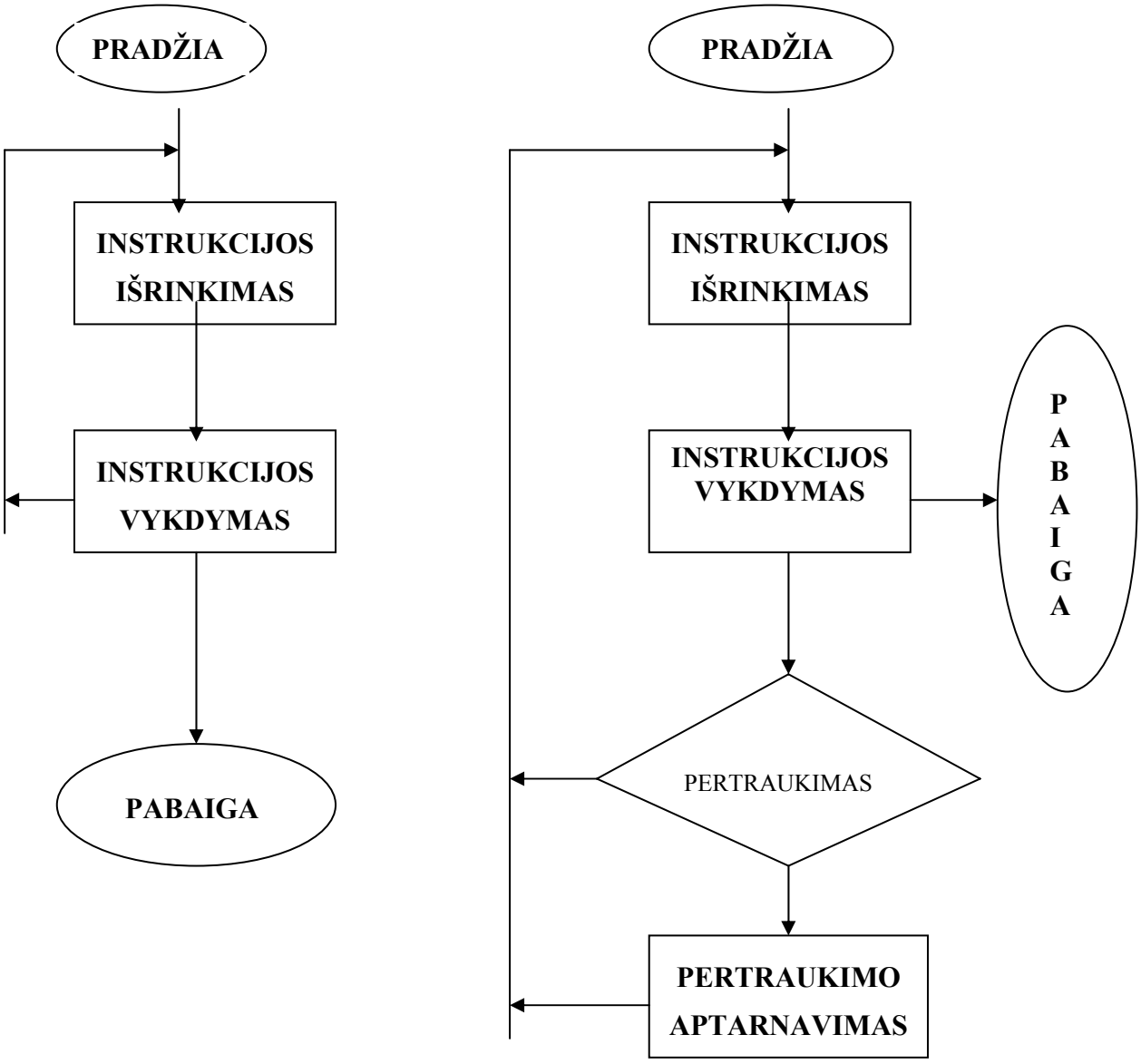
4.1. Supervizoriaus režimas

Procesoriaus persijungia į supervizoriaus (branduolio) režimą pertraukimais arba sisteminiais kreipiniais.

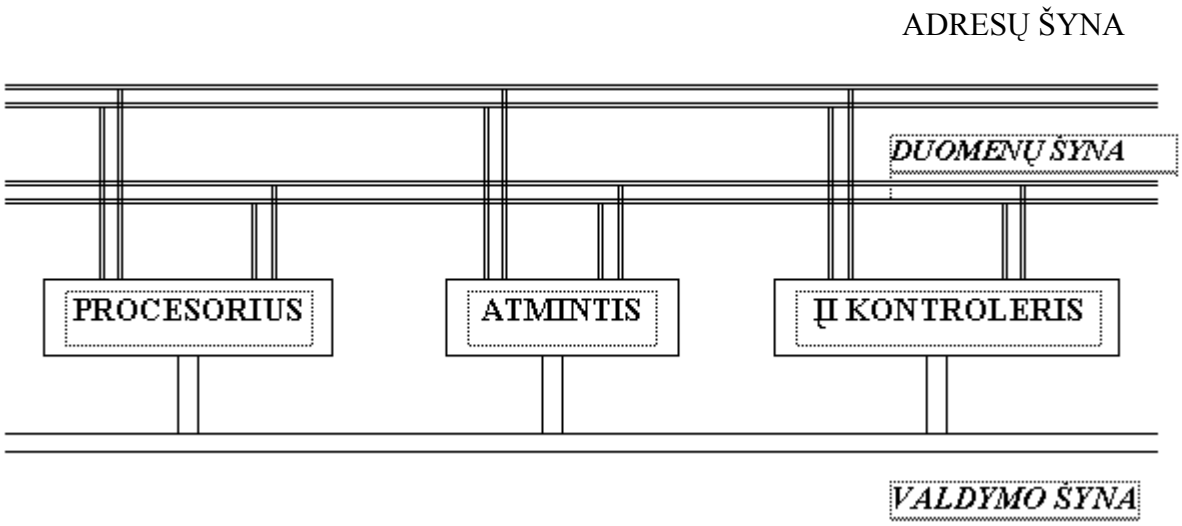
Pertraukimų sistema

Įgalina procesorių vykdyti keletą programų ir įvedimo – išvedimo operacijų vienu metu. Procesorius aptarnauja kiekvieną programą (operaciją) kai atsiranda poreikis.

Instrukcijų išrinkimo-vykdymo ciklas (fetch-execute cycle)



4.2. Adresų, duomenų ir valdymo šynos. Pertraukimų tipai. DMA



Pagrindiniai pertraukimų tipai

- **Įvedimo-Išvedimo** (*IO*). Generuojami ĮI įrenginių kontroleriais ir signalizuoja apie ĮI operacijų sėkmingą įvykdymą arba klaidos tipą.
- **Laiko** (*timer*). Generuojami vidiniu procesoriaus laikrodžiu, paprastai tam tikrais laiko intervalais, būtiniausių sisteminių darbų atlikimui.
- **Techninės įrangos klaidų** (*hardware error*). Pavyzdžiui, atminties pariteto klaida, maitinimo klaida.
- **Programiniai pertraukimai**. Vartotojų programų pranešimai apie savo klaidas OS (dalyba iš nulio, atminties apsaugos klaidas ir pan.) arba kreipiniai į OS servisus.

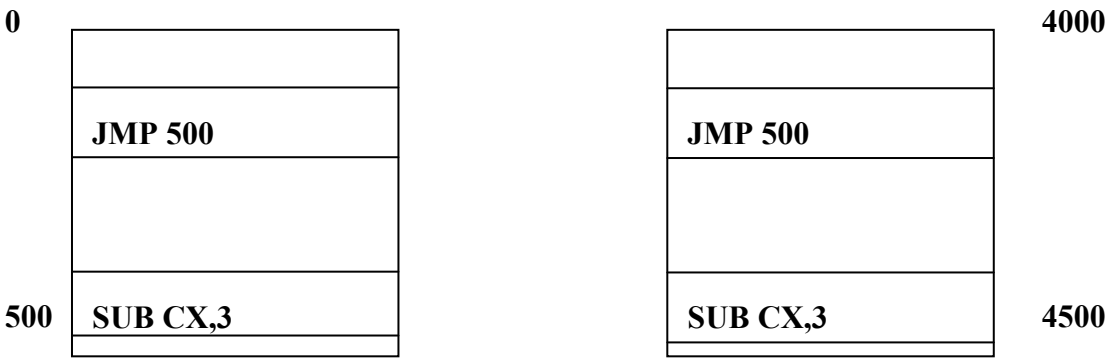
Tiesioginis priėjimas prie atminties (DMA)

- DMA įtaisas leidžia tiesiogiai persiusti duomenis iš atminties į įrenginius ir atvirkščiai. Nors DMA sustabdo procesorių (*cycle stealing*) duomenų siuntimo momentu, tai nėra pertraukimas, nes programos kontekstas nesaugojamas ir procesorius nieko neapdoroja.

Atminties adresavimas

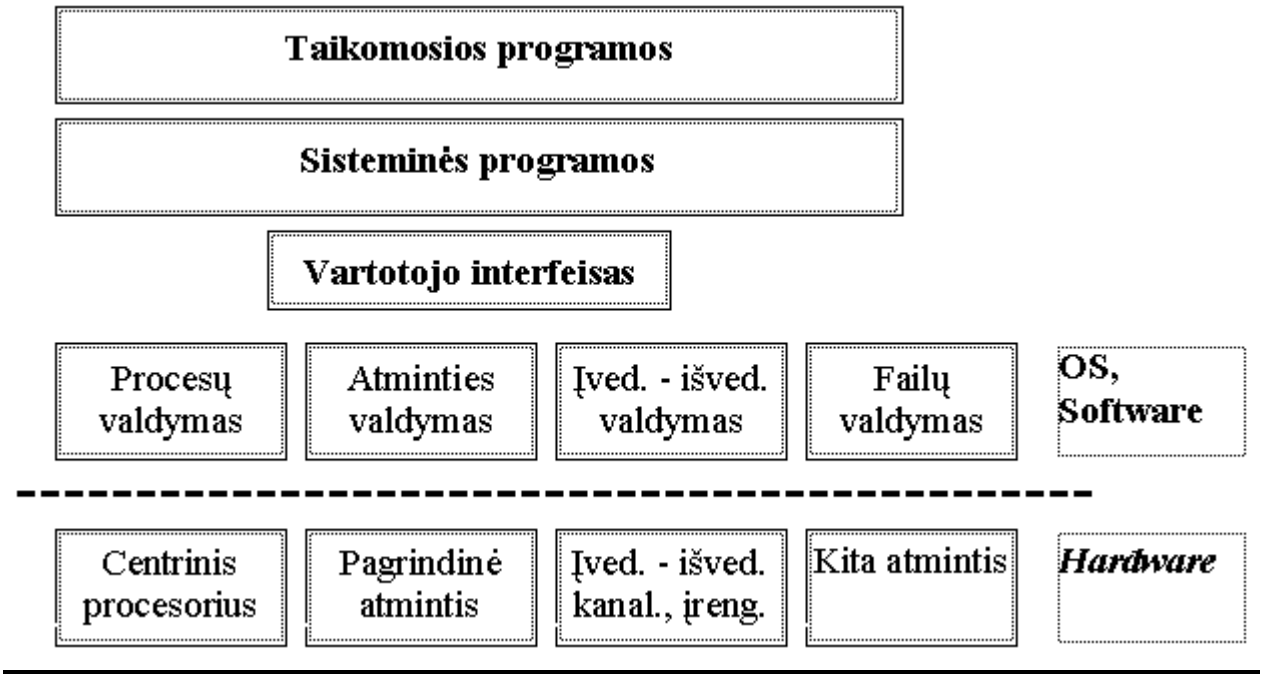
- Išrinkimo ciklo metu adresų šynoje talpinamas atminties adresas. Adresų šynos plotis, t.y. bitų skaičius (pvz., nuo Intel 80386: $\mathbf{2^{32}} = \mathbf{4}$ Gb).
- Prieštaravimas: Procesoriaus instrukcijų adreso lauko talpa, pvz., 16 bitų.
- Efektyvus adresas := bazinis adresas (base) + poslinkis (offset).

4.3. Programos talpinimas į atmintį (*memory relocation*)



- Pakrovėjas su atminties perskirstymu (*relocating loader*). Visų programos instrukcijų absoliutiniai adresai yra modifikuojami, kai programa kraunama į atmintį. *Palyginkite MS-DOS ir Windows.*
- Alternatyvi technika: santykiniai adresai ir tiesioginiai operandai.
- Programos adresų pakeitimo registras (*relocation register*). Veikia analogiškai baziniam registrui: prie kiekvieno programoje nurodyto adreso pridedamas šio registro turinys.

5. Apibendrinta OS struktūra



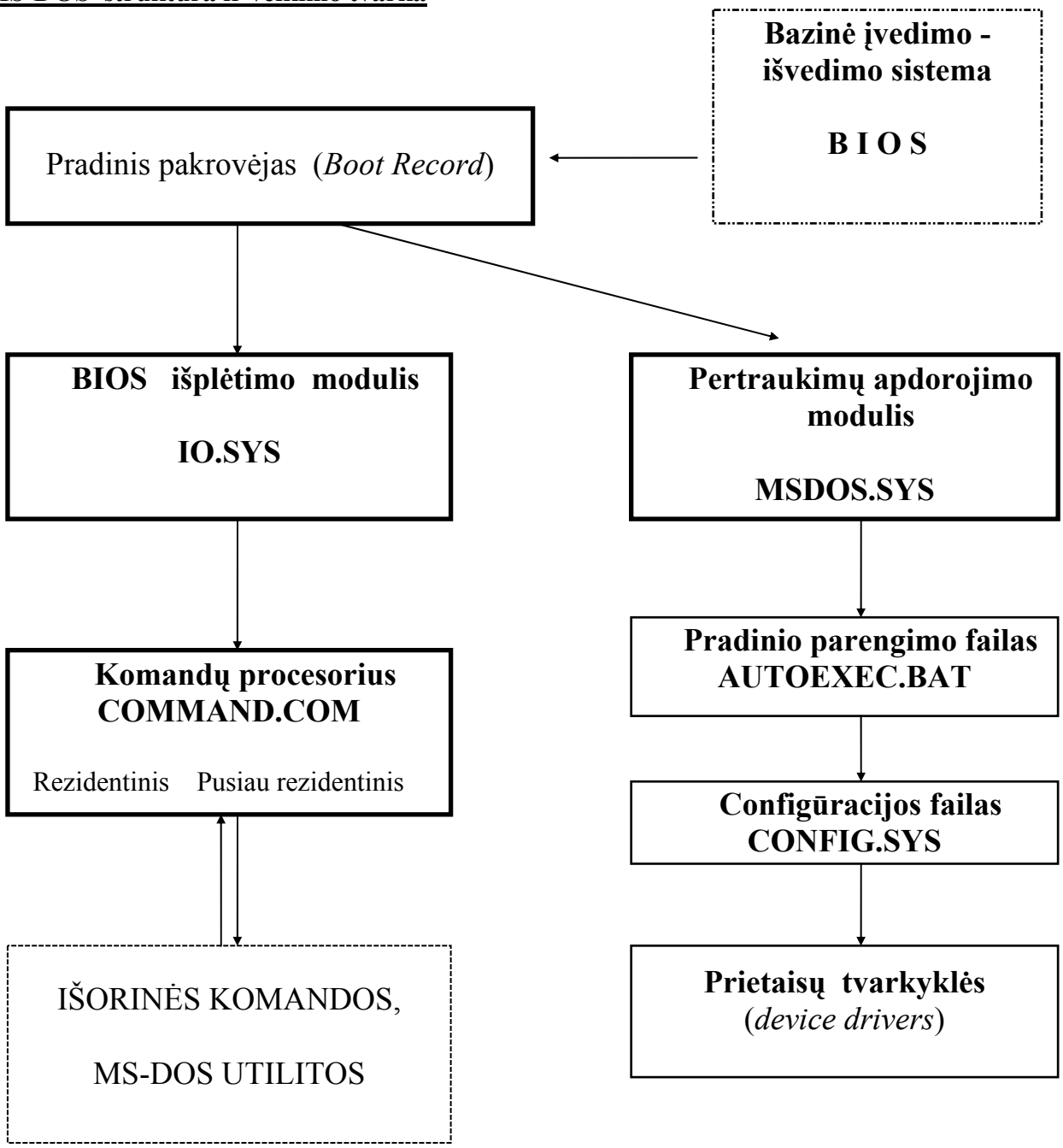
5.1. MS-DOS architektūra

Pagrindiniai OS uždaviniai

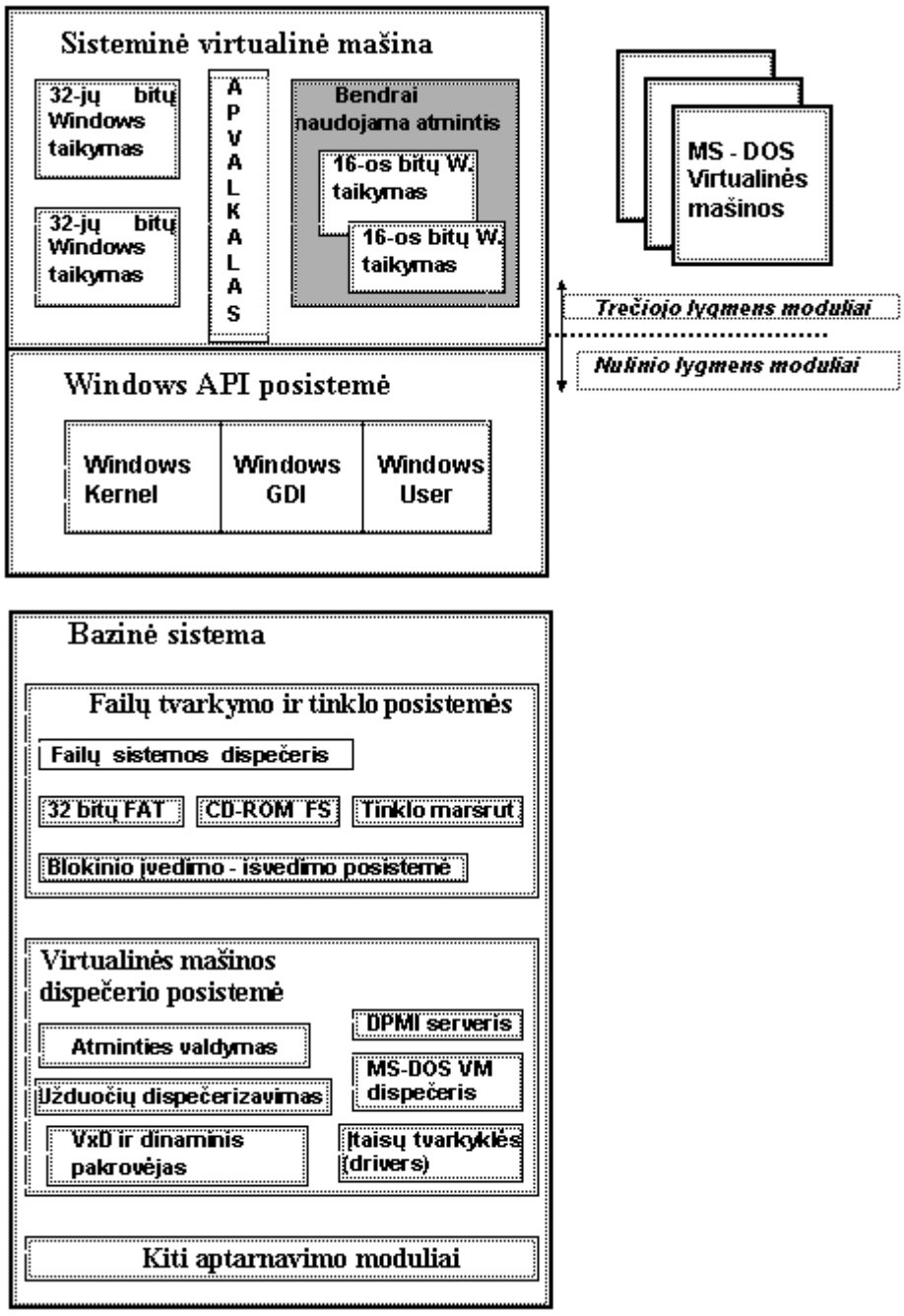
1. Aparatūros valdymas ir taikomųjų programų veikimo užtikrinimas: procesų kompiuteryje valdymas, atminties tvarkymas.

2. Vartotojo interfeiso užtikrinimas: įvedimo-išvedimo valdymas, failų sistemos palaikymas.

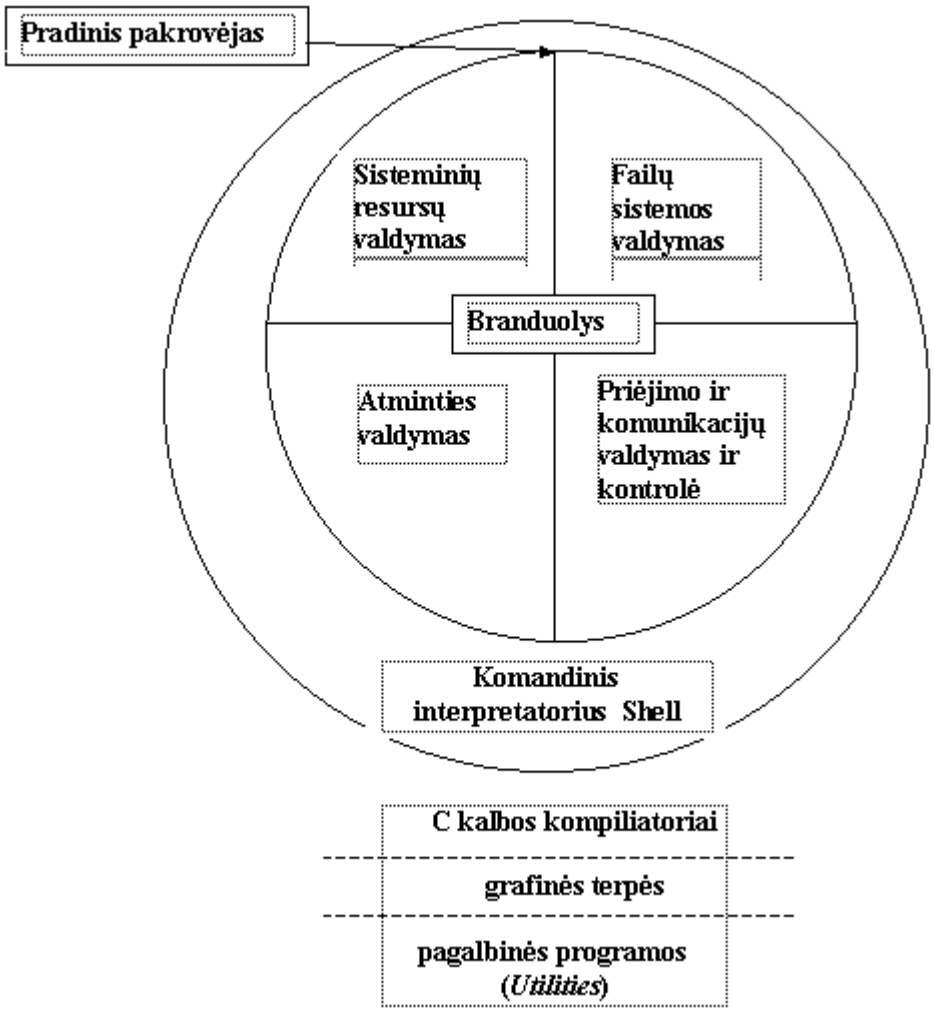
MS-DOS struktūra ir veikimo tvarka



5.2. Windows'95 architektūra



5.3. Unix OS architektūra



6. Procesų valdymas

6.1. Procesai, sub-procesai, virtualus procesai. Dispečerizavimo lygiai

Procesas – procesoriaus darbo vienetas. **Procesas – programos vykdymas.**

Procesas ≈ užduotis (*task*). *Multiprocessing* – reiškia vykdyti daug procesų. Procesas visiškai kontroliuoja procesorių (jis vykdo proceso instrukcijos) tol kol:
• procesas iškviečia sisteminę funkciją; • įvyksta pertraukimas.

Prieš vykdant savo kodą OS "keičia proceso kontekstą", t.y., saugo kompiuterio būseną. Tai papildomos procesoriaus laiko sąnaudos.

Thread (light-weighted process) – **sub-procesas**, kodo seka duoto proceso kontekste. Sub-procesai bendrai naudojasi proceso atmintimi, bet atskirai dispečerizuojami. Sub-procesai efektyviau naudoja procesoriaus laiką ir atminties resursus. Jie naudojami moderniose OS, paskirstytose ir *client-server* sistemose. Jose **realūs procesai** vaizduojami kaip **virtualūs procesoriai (VP)**, kuriuose vykdomi virtualūs procesai, t.y., realūs sub-procesai. VP turi prioritetus.

OS/2 ir Windows NT, 9x sistemose yra *thread* palaikymo mechanizmai. Ypač aktualus sub-procesų palaikymas daugiaprocesorinėse sistemose (Windows NT).

Sub-procesus taip pat galima įgyvendinti taikomosiose programose be OS palaikymo, tai daro kai kurios DBMS.

Sub-procesų trūkumai: jie konfliktuoja dėl bendrų resursų panaudojimo.

Dispečerizavimas (*scheduling*).

Procesai konfliktuoja dėl procesoriaus panaudojimo. Tad, OS turi nustatyti optimalią procesų seką ir išskirti kiekvienam iš jų tam tikrą procesoriaus laiko intervalą. *Scheduling* vykdomas tokiuose lygmenyse:

• Aukšto lygio (*job scheduling, long term*) dispečerizavimas nustato ar leisti startuoti naujam procesui.

• Tarpinio lygio (*intermediate scheduling, medium level*) dispečerizavimas sustabdo ar atnaujina proceso vykdymą.

• Žemo lygio (*processor scheduling, short term*) dispečerizavimas nustato kuriam paruoštam vykdymui procesui skirti procesoriaus laiką. Būtent šis lygmuo paprastai vadinamas "dispečeriu" (dispatcher).

6.2. Procesų gyvavimo ciklas (*life cycle*)

Pvz., UNIX sistemoje vartotojas, naudodamas *Shell* interfeisą, įveda programos failo vardą *prog1*.
1. *Shell* randa failą ir generuoja kreipinį į sistemą procesui sudaryti.

2. OS sukuria atmintyje PCB (*Process Control Block*).

3. Procesas *prog1* pervedamas į vykdymo (*running*) būseną.

4. Procesui *prog1* reikia nuskaityti duomenis iš disko. Generuojamas sisteminis kreipinys ir procesas *prog1* pereina į užblokuotą (*blocked*) būseną, nes turi laukti I/O operacijos pabaigos.
5. Tuo metu kitas vartotojas naudodamas *Shell* interfeisą įveda programos failo vardą *prog2*.
Kartojasi punktai 1-3 procesui *prog2*.

6. Pasibaigus I/O operacijai, procesas *prog1* norėtų atnaujinti savo vykdymą. Procesorius dar užimtas *prog2* vykdymu, tad *prog1* atsiduria paruošiamojoje (*ready*) būsenoje. OS suformuoja tokių procesų eilę.

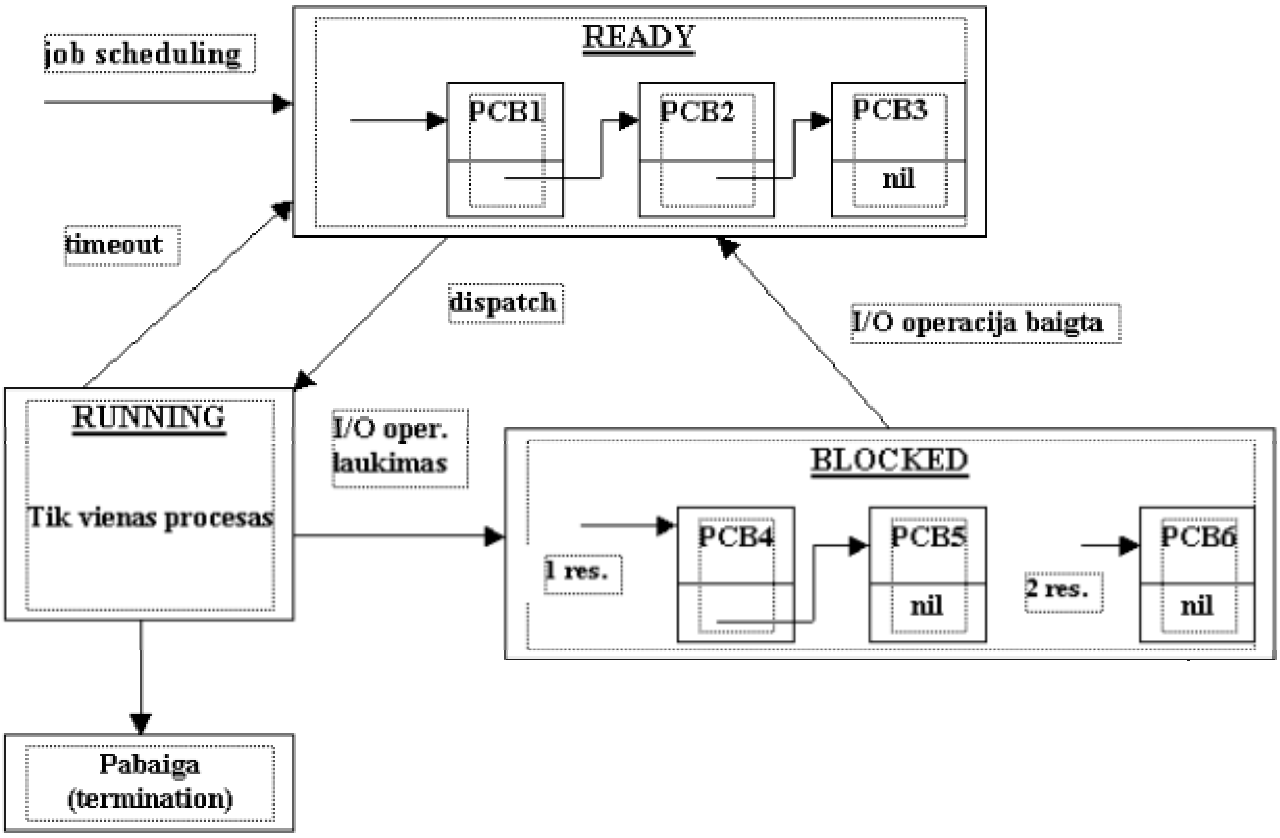
7. Dispečeris nusprendžia, kad pasibaigė procesui *prog2* skirtas laiko intervalas, ir perveda *prog2* į paruošiamąją būseną (ne užblokuotą). Tai vadinama *timeout*.

8. *Prog1* startuoja iš naujo, t.y., pereina į vykdymo (*running*) būseną.

9. Procesas *Prog1* baigia savo darbą ir pašalinamas iš sistemos.

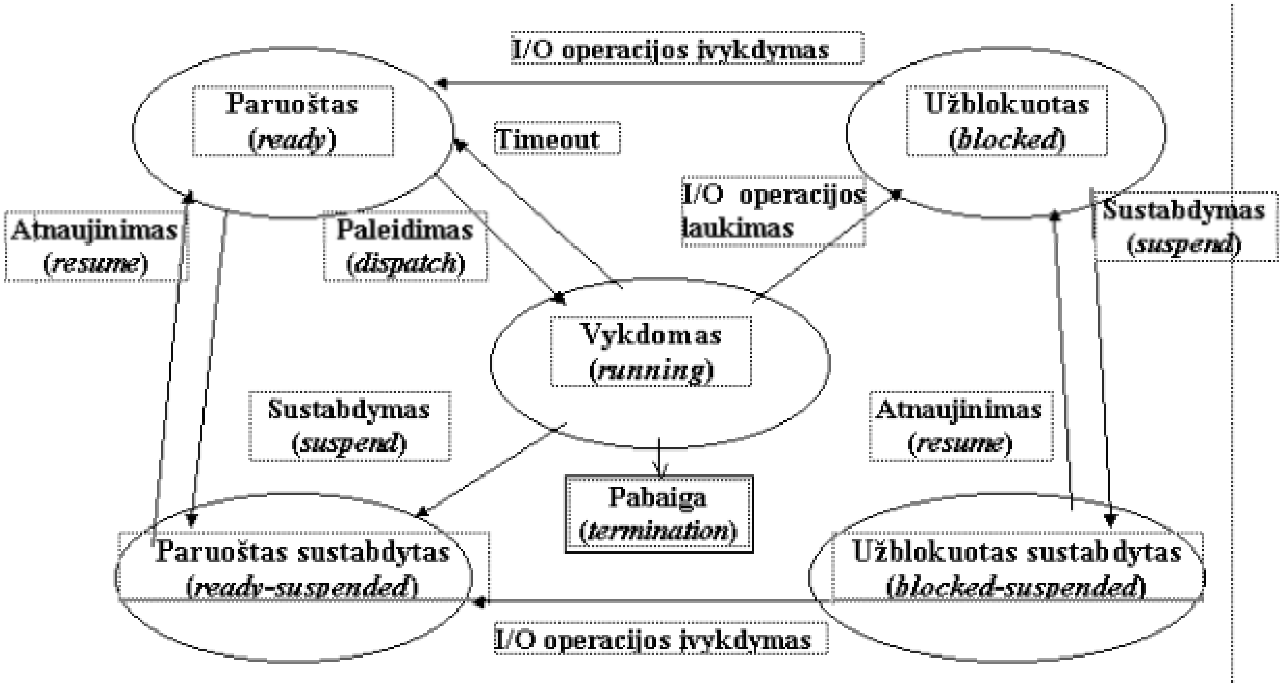
Proceso kūrimas susideda iš kodo talpinimo į atmintį ir PCB sudarymo. PCB ir kitose OS struktūrose yra aprašyti procesui skirti resursai: atmintis, išoriniai įrenginiai, disko erdvė, failai ir t.t. Procesai gali kurti kitus procesus; tada jie tampa *parent* procesais, o jų sukurti – *child.* Susidaro procesų hierarchinė struktūra (UNIX, OS/2). *Parent* procesas gali pasibaigti ankščiau nei *child* procesas.

Procesų būsenų modeliai ir diagramos.



6.3. Penkių būsenų modelis

• Papildomi tarpinio lygio veiksmai: proceso **sustabdymas** (*suspending*), kai proceso vykdymas atidėdamas (pvz., dėl atminties puslapių apsikeitimo) iki jo **atnaujinimo** (*resuming*) vartotojo arba sistemos.



Pagrindiniai dispečerizavimo uždaviniai:

- siekti maksimalaus sistemos produktyvumo efektyvumo;
- vykdyti visus vartotojo procesus, atsižvelgiant į jų prioritetus ir reikalavimus sistemai;
- skirti procesams atitinkamus laiko kvantus ir reaguoti į vartotojo veiksmus;
- užtikrinti sistemos stabilumą.

Dispečerizavimo kriterijai:

- užduoties prioritetas;
- užduoties tipas (procesoriaus ar I/O pagrįstas);
- užduoties klasė:
  komandinė (*batch*), interaktyvi (*on-line*), realaus laiko (real-time);
- reikalavimai resursams (procesoriaus laikas, atmintis, I/O);
- jau panaudotų resursų apimtis;
- laukimo laikas.

Dispečerizavimo metodai:

FIFO (FCFS) pirmo uždavinio; *SJF,* S*hortest Job First*-trumpiausio uždavinio;
RR, Round Robin - žiedo-karuselės;
SRT, Shortest Remaining Time - trumpiausio likusio laiko;
HRN, Highest Response Ratio Next - aukščiausio dinaminio prioriteto;
MFQ, Multilevel Feeedback Queues - eilių su grįžtamaisiais ryšiais rinkinio.

6.4. Žemo lygio dispečerizavimas

Žemo lygio dispečerizavimas gali būti:

- uždraudžiantis ("*preemptive*");
- neuždraudžiantis ("*non-preemptive*");
- kooperatyvus (*non-preemptive*), pvz., Windows 3.x taikymų ciklas.

Žemo lygio dispečerizavimo metodai:

- *FIFO (FCFS)* - pirmo uždavinio. Tai *non-preemptive* metodas, duodantis privalumą dideliems uždaviniams. Sant =(Laukimo laikas)/(Vykdymo laikas)

Užduotis	Vykdymo laikas	Laukimo laikas	Sant
1	2	0	0
2	60	0	0.03
3	1	62	62
4	3	63	21
5	50	66	1.32

Procesoriaus pagrįsti uždaviniai sąlygoja I/O tipo uždavinių prastovą.

- *SJF,* S*hortest Job First*-trumpiausio uždavinio. Čia prioritetas trumpiems uždaviniams (minimalaus vykdymo laiko).

Užduotis	Vykdymo laikas	Laukimo laikas	Sant
3	1	0	0
1	2	1	0.5
4	3	3	1.0
5	50	6	0.1
2	60	56	0.9

Pasižymi mažiausiu vidutiniu laikymo laiku. Tinka paketiniam režimui, bet pageidauja vykdymo laiko įvertinimo.

Dideli procesai gali užstrigti eilėje ("*starvation*").

- *SRT, Shortest Remaining Time* - trumpiausio likusio laiko. Tai SJF metodo *preemtive* versija, tinkanti paketiniam režimui. Reikalauja vykdymo laiko ir įvertinimo ir likusio laiko apskaičiavimo.

- *HRN, Highest Response Ratio Next* - aukščiausio dinaminio prioriteto. SJF metodo modifikacija leidžianti išvengti didelių procesų užstrigimo, pvz.,

Prioritetas =(laukimo laikas + vykdymo laikas) / vykdymo laikas.

• *RR, Round Robin* - žiedo-karuselės. Aparatinis *preemptive* metodas: naudojant kompiuterio taimerį įvedamas fiksuotas laiko intervalas, skirtas kiekvienam procesui - kvantas. Kvanto dydžio nustatymas, tipiškai 10..20 mls.

• *MFQ, Multilevel Feeedback Queues* - eilių su grįžtamaisiais ryšiais rinkinys, kuriame paskutinė eilė veikia RR principu. Adaptyvi schema, vertinanti procesų vykdymo istoriją, jų tipą ir dydį.

6.5. Procesų konkurencija

Procesai konkuruoja dėl bendrų resursų panaudojimo ir bendrauja tarpusavyje. Procesai gali būti: visiškai nepriklausomi, nepriklausomi bet tarpusavyje susiję, kooperuojantys ir **konkuruojantys** procesai.

• Resursai: fiziniai ir loginiai; atstatomi, nuosekliai atstatomi ir neatstatomi. • Resursų panaudojimo problemos: nuosekliai atstatomi resursai gali būti paskirti tik vienam procesui vienu metu, todėl reikalingas procesų ir jų resursų tarpusavio atskyrimas (*mutual exclusion*). Toks resursų atskyrimas (*mutual exclusion*) sudaro pagrindą procesų užblokavimui (aklavietei - *deadlock*). Procesas yra užblokuotas jeigu jis laukia įvykio, kuris nekada neįvyks. Pvz., procesai P1, P2 ir P3 naudoja resursus R1, R2, R3. Vienas iš galimų užblokavimo variantų būtų toks:

Procesas	Naudojamas resursas	Reikiamas resursas
P1	**R1**	**R2**
P2	**R2**	**R3**
P3	**R3**	**R1**

• Procesų komunikacijos būdai: bendrai naudojama atmintis, bendro naudojimo failai, pranešimų siuntimas, įvykių sinchronizavimas.

• **Kritinė sekcija** (KS) – proceso kodo dalis, reikalaujanti bendrų resursų panaudojimo vienu metu. Kad išvengtų užblokavimo, du susiję procesai negali būti savo kritinėse sekcijose tuo pačiu metu.

Procesų atskyrimo metodai.

Sisteminius resursus centralizuotai skirsto OS, bet loginių resursų atskyrimui reikia tam tikrų specialių metodų. Tai gali būti programinės konstrukcijos, techniniai (architektūros) sprendimai ir OS priemonės.

Pvz., du procesai A ir B bendrai naudoja kokį nors resursą ir turi kritines sekcijas. Problema: paprasti sprendimai neefektyvus, nes atskiri procesų įvykiai neapibrėžti laike. Tegul loginis kintamasis *res_atviras* rodo resurso būseną ir pradžioj yra *true.* Kiekvienas procesas valdo *res_atviras* tokiu būdu:

	Procesas A	***Procesas B***
while not *res_atviras*	tikrina *res_atviras*	
do *laukti;*		tikrina *res_atviras*
res_atviras := false;	uždaro resursą	
		uždaro resursą
{ KS }	kritinė sekcija	kritinė sekcija
res_atviras :=true;		

Kita pateikto sprendimo problema - procesoriaus laiko atžvilgiu labai neefektyvus ciklas ***while.*** Tokia situacija vadinama "*busy waiting*".

• Programinių konstrukcijų sprendimai. Petersono algoritmas (1981) naudoja kiekvienam procesui loginius požymius *flagI* ir bendrą sveikąjį kintamąjį *turn*, kurių pradinės reikšmės *false* ir *1*.

Procesas A
flag0 := true;
turn := 1;
while *flag1* ***and*** *turn=1*
do laukti;
{kritinė kodo sekcija}
flag0 := false;

Procesas B
flag1 := true;
turn := 0;
while *flag0* ***and*** *turn=0*
do *laukti;*
{kritinė kodo sekcija}
flag1 := false

Tokiu būdu galima aptarnauti N procesus turint *N flags* ir *turn : 0 .. N-1.*

• Techniniai sprendimai. Paprasto sprendimo problema -reikšmės *res_atviras* patikrinimas ir nustatymas - tai dvi atskiros komandos, todėl kito proceso komandos gali būti įvykdytos tarp jų. Čia įvedama procesoriaus komanda ***test-and-se***t, kuri vykdo tiek patikrinimą, tiek nustatymą vienoje mašininėje komandoje, tad ir negali būti pertraukiama. Komanda naudoją sveikąjį kintamąjį *jungiklis*, kuris pradžioje lygus 0. Jei šis kintamasis yra 0, komanda keičia jį į 1 ir gražina reikšmę *true*, priešingu atveju gražina *false.*

while test_and_set *(jungiklis)*
do *laukti;* {neefektyvus ciklas "*busy waiting*"}
{ KS }
jungiklis :=0;

• Semaforai *"semaphore"* (Dijkstra, 1965). Būna binariniai ir skaičiuojantys.

Semaforas *s* - teigiamas sveikasis kintamasis su kuriuo galima atlikti dvi specialias operacijas: ***wait*** (dar vadinama ***P***) ir ***signal*** (arba V). Įėjimas į KS kontroliuojamas ***wait*** operacija; apie išėjimą iš KS signalizuoja ***signal.***

wait(s):
if *s > 0*
then *s := s - 1*
else *{užblokuoti kviečiantį procesą};*
wait *(s);{* ***KS*** *} ;* ***signal*** *(s)*

signal(s):
if *{yra užblokuotas procesas (-ai)}*
then *{pradėti kurį nors procesą}*
else *s := s + 1;*

Semaforų operacijų naudojimas: ***wait*** *(s);* *{* ***KS*** *} ;* ***signal*** *(s)*

Kiekvienam semaforui sistema sudaro jo laukiančių procesų eilę. OS taip pat garantuoja, kad šios operacijos yra nepertraukiamos. Kitas privalumas - OS perveda laukiantį procesą į, pvz., būseną "sustabdytas", t.y., kol vyksta laukimas, procesoriaus laikas nenaudojamas.

• Gamintojo – vartotojo problema.

Procesas - gamintojas generuoja duomenis ir saugo baigtiniame buferyje (masyve 1..N). Procesas – vartotojas paima duomenis iš buferio ir tam tikru būdu apdorojo. Pavyzdys: informacijos spausdinimas.

Aptarnavimo disciplina turi garantuoti:

- procesų tarpusavio atskyrimą;
- galimybę nustatyti buferio užpildymą ir uždrausti gamintojui tolesnį rašymą į pilną buferį; gamintojas turi laukti, kol buferyje atsiras laisvos vietos;
- galimybę nustatyti momentą kai buferis tampa tuščiu bei uždrausti vartotojui tolesnį skaitymą iš buferio; vartotojas turi laukti, kol buferyje atsiras duomenų.

Galimas sprendimas: naudojami tokie trys semaforai

Semaforas	Semaforo paskirtis	Pradinė reikšmė
Free	Tarpusavio atskyrimas	1
Space	Laisva buferio atmintis	N
Data	Duomenų apimtis buferyje	0

Gamintojo ir vartotojo procesų struktūra.

Gamintojas:
Paruošti elementą Procesas gamina informaciją
Wait (space) Jei buferis pilnas, laukti **space** signalo
Wait(free) Jei buferis naudojamas, laukti **free** signalo
Pridėti elementą į buferį
Signal (free) Signalizuoti, kad buferis nebenaudojamas
Signal (data) Signalizuoti, kad buferyje yra duomenų

Vartotojas:
Wait (data) Sulaukti nors vieno elemento buferyje
Wait(free) Jei buferis naudojamas, laukti **free** signalo
Paimti elementą iš buferio
Signal (free) Signalizuoti, kad buferis nebenaudojamas
Signal (space) Signalizuoti, kad buferyje yra laisvos vietos
Panaudoti elementą Vartotojas naudoja informaciją

Data ir Space yra skaičiuojantys semaforai, Free – binarinis.
Operacijos *Wait* ir *Signal* atitinkamai mažina ir didina vienetu semaforų reikšmes.

• Problemos.
Programos, kuriose naudojami semaforai, yra sunkiai derinamos. Tipinė klaida - amžinasis *Wait* ciklas ir *Signal* nebuvimas. Alternatyva – aukšto lygio dispečerizavimo sprendimai įgyvendinti programavimo kalbose, pvz., *monitoriai.*
Programavimo kalbos: *Concurrent Pascal, ADA, Concurrent C.*

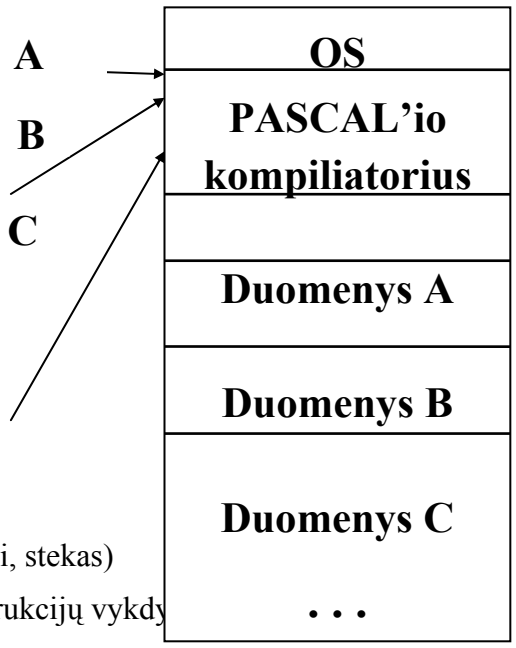
Procesų koncepcija: ***procesą vykdo procesorius***

Programos sąvokos nepakanka:
• atmintyje vienu metu g. b. keletas tos pačios programos kopijų;
• procesorius gali vykdyti vieną programą kelis kartus vienu metu.

Programa, kurios vienas egzempliorius naudojamas kelių procesų vykdymui vienu metu, vadinasi *"**reenterabili".***

Reenterabilumo sąlygos:

• programos kodas nesikeičia jos vykdymo metu (invariantas)
• kiekvienas vykdymas naudoja vienodai pažymėtus duomenis skirtingose atminties vietose.



Proceso kontekstas:

• procesoriaus kontekstas (PSW ir registrai)
• proceso atributai (vardas, prioritetas, teisės)
• atminties kontekstas (kodo ir duomenų segmentai, stekas)

Proceso trasa - jo konteksto būsenos seka po instrukcijų vykdy

6.6. Procesų valdymas UNIX sistemoje

• Procesas - bazinė UNIX koncepcija.
• Sistemos branduolys tenkina procesų poreikius.
• Proceso sukūrimas - tai virtualinės mašinos, turinčios savo adresų erdvėje kodo ir duomenų segmentus, sudarymas. PID - proceso identifikatorius.

Sistemos pakrovimas

• Branduolys inicijuoja pradinį procesą, kurio PID=0 (*shed* arba *swapper*).
• Procesas su ID=0 paleidžia procesą 1 (*init*), kuris tarnaus visų sistemoje esančių procesų protėviu. Procesas 1 paleidžia *shell'*o procesus, kurie suteikia vartotojams galimybę užsiregistruoti (*login*) ir inicijuoti savo procesus.
• Kiekvienam vartotojui atskirai paleidžiamas sisteminis procesas *getty*, kuris inicijuoja vartotojo *login* ir *Shell (sh).* Procesas1 paleidžia ir *daemon* procesus.
• Visi procesai specifikuojami faile */etc/initab,* vartotojai - */etc/passwd.*

Vartotojo procesų sukūrimas

• Visi procesai UNIX sistemoje inicijuojami sisteminiu kreipiniu *fork(),* kuris sudaro dvi paleidžiamojo proceso kopijas (protėvio *parent* ir palikuonio *child*) su skirtingais ID. Paprastai *child* procesas keičia tėvo kodą į savo, naudodamas sisteminį kreipinį *exec,* pavyzdžiui:

```
main()
{
        int status;
        if (fork() == 0)
```

exec (“/bin/manopr”,”manopr”,0);

wait (&status);

}

Parent procesas lauks *child* proceso vykdymo pabaigos, naudodamas sisteminį kreipinį *wait.*
• Analogiškai veikia ir sistemos “apvalkalas” *Shell,* aptarnaudamas vartotojus. Pradžioje jis naudoja *fork* ir tokiu būdu sukuria dar vieną savo kopiją. Toliau *child Shell* kreipiasi į reikiamą programą naudojant *exec* funkciją. Programos kodas perdengia *Shell* kodą ir yra vykdomas.

Procesų pabaiga (termination)

• Procesai protėviai laukia procesų palikuonių vykdymo pabaigos naudodami sisteminį kreipinį *wait.* Procesai baigia darbą naudodami sisteminį kreipinį *exit*

• Procesai gali pasibaigti normaliai ar nenormaliai, t.y., ankščiau laiko. Kiekvienas procesas identifikuojamas PID ir PPID. *Init* procesas - tai PPID=1 procesų protėvis.

Procesų valdymo komandos (LINUX)

• **Procesų paleidimas:**

at - vykdo komandą tam tikru laiku

at 2:00

*lp /usr/ataskaitos/**

echo “pranešimas dekanui” | mail -s“Ataskaita paruošta” dekanatas

batch - OS nusprendžia, kada vykdyti komandas, surašytas po *batch*

cron ir ***crontab*** - programų paleidimas pagal tvarkaraštį, surašytą vartotojų failuose kataloge *usr/spool/cron/crontab.* Vartotojo ir failo vardai sutampa.

• **Procesų monitoringas:**

who ir ***w*** - vartotojų sąrašo išvedimas ir naudojamų resursų pateikimas.

nice ir ***renice*** - procesų prioritetų nustatymas prieš vykdant procesą ir proceso vykdymo metu. Kiekvienam procesui priskiriamas prioritetas, LINUX sistemoje 0<= pr <=20; aukščiausią prioritetą atitinka 0, o sisteminiai procesai gali turėti ir neigiamas prioriteto reikšmes. Paprasti procesai turi standartini prioritetą = 10.
nice -5 1998 - mažina prioritetą proceso su PID=1998;

nice --10 lp ataskaita - didina spausdinimo proceso prioritetą iki maksimumo.

ps - informacijos apie procesų būsenas pateikimas. Paprastai pateikiama:

PID - proceso ID;

PPID - proceso protėvio ID;

TTY - terminalas, iš kurio paleistas procesas;

TIME - proceso vykdymo laikas;

CMD - įvykdytos komandos vardas

• **Procesų pabaiga:**

kill - baigia proceso vykdymą.

kill 1998 102 - baigia procesų 102 ir 1998 darbą,

kill -signal PID - pvz., *kill -9 1998* staigus proceso baigimas.

kill 0 - foninių procesų pabaiga.

nohup - leidžia procesui dirbti po vartotojo išėjimo iš sistemos.

Semaforai ir įrašų blokavimas UNIX sistemoje

UNIX suteikia galimybę kurti ir valdyti semaforus. Tam naudojami kreipiniai į tokias UNIX funkcijas:

semget – sukuria semaforų rinkinį, t.y., nustato jų skaičių, identifikatorių, bei kitus valdymo parametrus;

semctrl – inicializuoja semaforus, analizuoja jų būsenas, šalina semaforų rinkinį ir pan.;

semop – atlieka operacijas su semaforais naudojant duomenų struktūrą sembuf, susidedanti iš 3-jų reikšmių:

sem_num – tam tikro semaforo indeksas masyve; naudojamas operacijoms *wait* ir *signal* realizuoti;

sem_op - sveikojo tipo reikšmė, naudojama semaforui modifikuoti;

sem_flg – požymis (flagas) operacijai nustatyti.

UNIX funkcija *fcntl* naudojama įrašų blokavimui. Kadangi sistema neturi informacijos apie loginį taikomosios programos įrašą, ji naudoja fizinius disko sektorių adresus ir vartotojo pateiktus įrašų ilgius. Tokiu būdu galima blokuoti įrašus naudojant santykinius (nuo dabartinės pozicijos faile) adresus.

Aklavietės (*deadlocks*). Pavyzdžiai ir atsiradimo sąlygos.

- Dar karta apie tarpusavio atskyrimo (*mutual exclusion*) būtinybę.

Pavyzdys 1. Paskutinio bilieto į reisą TE451 (Vilnius-Londonas) pardavimas vienu metu Jonui (Klaipėdoje) ir Petrui (Šiauliuose). Abejų agentūrų operatoriai formuoja užklausą į centrinę duomenų bazę (DB), esančią Vilniuje, ir gauna atitinkamą įrašą į savo vietinį kompiuterį. Petro operatorius pirmas rezervuoja vietą ir atnaujina DB, tačiau Jonas irgi gauna rezervavimą ir jo operatorius dar kartą perrašo DB įrašą. Kiekvienas klientas mano, kad jo bilietas yra užsakytas, bet, iš tikrųjų, bilietą gauna tik Jonas. Be abejo, realioje sistemoje tokios situacijos neturėtų būti – reikalingas procesų tarpusavio atskyrimas.

Pavyzdys 2. Patobulintoje sistemoje bet kurį užklausa blokuoja įrašą. Dabar Petras rezervuoja bilietą į Londoną, o Jono operatorius negauną priėjimo prie jo. Tačiau, Petras norėtų užsisakyti ir atbulinį bilietą į reisą TE452 ir tol kol jį gaus, laikys jau rezervuotą bilietą. Tuo metu Jono operatorius, negavęs TE451 bilieto, užsisakė paskutinį bilietą į atbulinį reisą TE452 ir laikys jį tol kol negaus tiesioginio bilieto į TE451. Vėl aklavietės situacija, kuri susidaro dėl tam tikros įvykių tvarkos jau patobulintoje sistemoje.

- Aklavietės būtinos sąlygos.
  - Tarpusavio atskyrimas, t.y., vienu laiko momentu, vienam procesui vienas resursas. Bendro naudojimo resursas neiššaukia aklavietės.
  - Resurso (su-) laikymas, t.y., procesas laiko vieną ir laukia kitų resursų. Kai procesas iš karto gauna visus jam būtinus resursus, jam nėra ko laukti ir aklavietė neatsiras.
  - Nėra priverstinio resursų išlaisvinimo (*no preemption*)
- Aklavietės pakankama sąlyga – ciklinis resursų laukimas.

Aklavietės problemos sprendimo būdai.

Profilaktika (prevencija) – taikyti tokia resursų skirstymo strategiją, kad aklavietė negalėtų įvykti (t.y. nepatenkinti nors vieną sąlygą).

Vengimas – neleisti tokį konkretų resursų skirstymą, kuris veda į aklavietės būseną.

Sekimas – susidoroti su aklaviete, kai ji jau įvyko.

- Aklaviečių profilaktikos problema
  - Tarpusavio atskyrimo paprastai neįmanoma išvengti (plg. su dvigubu bilieto užsakymu).
  - Resurso (su-) laikymas. Jei procesas iš karto gauna visus jam būtinus resursus, aklavietė neatsiras, bet atsiranda kitos problemos:
    - procesas gali labai ilgai laukti visų resursų išlaisvinimo, nors jo paleidimui ne visi jie reikalingi;
    - procesas gali labai ilgai laikyti visus resursus be realaus jų naudojimo;
    - kai procesas laikinai atleidžia resursą, kitas procesas gali jį "pagrobti".
  - Priverstinį resursų išlaisvinimą paprastai vykdo operacinė sistema, suteikiant aukštesnio prioriteto procesui reikiamą resursą. Toks būdas netinka nuosekliai atstatomiems resursams.
  - Ciklinio resursų laukimo galima išvengti, įvedant tam tikra resursų pareikalavimo tvarką. Pvz., spausdintuvas->kompaktinis diskas->kietas diskas. Kai procesas laiko spausdintuvą, jis gali užimti ir diskus, bet užimant diską, procesas negali pretenduoti į spausdintuvą. Tam jis turi išlaisvinti diską. Nesunku matyti, kad taip išvengiamas ciklinis laukimas, nors šiš būdas ir nepasižymi dideliu resursų naudojimo efektyvumu.
- Aklaviečių vengimo problema.

Aklaviečių profilaktika reikalauja daug kompiuterinio laiko ir kitų resursų. Išvengti aklaviečių galima analizuojant visus resursų pareikalavimus ir nustatant ar susidaro ciklinio laukimo sąlygas. Tai vėl gi reikalauja per daug laiko.

Kitas būdas – leisti įgyvendinti tik tokius pareikalavimus, kurie potencialiai nepavojingi. Vienas iš tokių žinomų metodų (Dijkstra !) – bankininko algoritmas. Analogija: bankas duoda kreditus tik tuo atveju, jei jis galės ateity suteikti kreditus ir kitiems klientams (prognozuojant jų poreikius).

Vengimo algoritmai reikalauja reikiamų resursų deklaravimo. Iš tokių deklaracijų sistema ir nustato ar reikalavimai yra "saugūs", t.y. neveda į aklavietę.

Pavyzdys. Tarkime turime 3 procesus, P1, P2 ir P3 ir 10 vienodo tipo sisteminių resursų (pvz., diskus). Maksimalių reikalavimų deklaracijos yra tokios:

Procesas	Max poreikis	Dabartinis resursų užimtumas
P1	**8**	**3**
P2	**5**	**1**
P3	**8**	**2**

Laisvų resursų skaičius – 4.

Bendras max poreikis (21) gali būti didesnis už esamą. Svarbu, ar atsirastų tokia resursų skirstymo kombinacija ir tvarka, kad visi procesai galėtų sėkmingai pasibaigti. Tokia seka galėtų būti, pvz.,: P2 (->5) P1 (->8) P3.

Kita situacija:

Procesas	Max poreikis	Dabartinis resursų užimtumas
P1	**9**	**3**
P2	**5**	**1**
P3	**8**	**2**

Laisvų resursų skaičius – 4.

Skirtumas tik vienas - P1 reikia vienu resursu daugiau, bet tai sudaro potencialiai "nesaugią" situaciją, kurios vengimo algoritmai ir neleidžia.

Vengimo metodai yra efektyvesni už profilaktikos metodus, bet turi savo apribojimus:

- kiekvienas procesas turi deklaruoti savo poreikius, kas nerealu interaktyviose sistemose;
- vengimo algoritmas turi būti vykdomas visada kai atsiranda naujas resursų pareikalavimas. Sistemoje, kurioje daug vartotojų, tai sunkiai įgyvendinama.

• Aklaviečių sekimas reikalauja periodinio ciklinio laukimo atpažinimo procedūros paleidimo. Be nustatymo reikia įveikti aklavietę ir atstatyti sistemos veikimą. Tiesioginis būdas, t.y., iš karto panaikinti visus užblokuotus procesus, gali būti pakeistas nuosekliu procesų uždarymu tol kol aklavietė nebus panaikinta.

Naikinamo proceso (aukos) pasirinkimas – tai atskira problema. Paprastai OS leidžia ir vartotojui naikinti užblokuotus procesus.

Apibendrinimas. Praktikoje naudojamos įvairios išvardintų metodų kombinacijos. Administratoriui taip pat svarbu numatyti periodinį duomenų saugojimą bei situacijos patikrinimą, suplanuoti atšaukimo (*roll-back*) ir atstatymo (*recovery*) procedūras.

6.7. Procesų komunikacijos metodai. Signalai

Procesų komunikacija

Šiuolaikinėse OS naudojamas platus komunikacijos metodų spektras:

- Signalai;
- Bendro naudojimo failai, arba vamzdžiai (*pipes*);
- Pranešimų siuntimas;
- Bendro naudojimo atmintis
- DDE – dinaminis duomenų apsikeitimas;
- OLE, ActiveX – objektų surišimas;
- Semaforai (kaip procesų sinchronizavimo priemonė).

• **Signalai.**

Primityvi komunikacijos forma; paprastai gaunami signalai apie nukreipimą nuo normalios situacijos. Signalai yra panašūs į pertraukimus: proceso vykdymas pertraukiamas ateinančiam signalui apdoroti. Paprastai signalai siunčiami iš OS branduolio taikomajam procesui ir proceso veikimas yra nutraukiamas. Kai kurie signalai gali būti siunčiami iš vieno taikomojo proceso kitam.

Pavyzdys 1. CTRL-C (arba DEL) klavišų kombinacija vykdomajai programai nutraukti. Čia, branduolys gauna ir atpažįsta pertraukimo kodą bei siunčia signalą SIGINT visiems duoto terminalo procesams. Kai taikomasis procesas gauna SIGINT, jis, pagal nutylėjimą, nutraukia vykdymą. Galima užprogramuoti programą taip, kad jos vykdymas nebūtų baigiamas SIGINT signalu (pvz., pats Shell irgi gauna SIGINT bet jo vykdymas nėra nutraukiamas.

Pavyzdys 2. Kai vykdomojoje programoje atsiranda perpildymo arba dalybos iš nulio situacija, branduolys gauna aparatūros pertraukimą, į kurį reaguoja siųsdamas taikomajam procesui SIGFPE signalą, kuris ir nutraukia proceso veikimą.

UNIX sistemos bendrųjų signalų sąrašas.

Signalo vardas	Trumpas aprašymas
SIGHUP	“Hang-up”, siunčiamas procesams, kai terminalas baigia darbą.
SIGINT	Terminalo inicijuotas pertraukimas.
SIGQUIT	Terminalo pertraukimas, iššaukiantis atminties “dump’ą”.
SIGILL	Neleistinos komandos vykdymas.
SIGTRAP	Trasavimo signalas, naudojamas UNIX programų derintojų.
SIGFPE	Slenkančio kablelio operacijų klaida.
SIGKILL	Siunčiamas vieno proceso, kad pabaigtų kitą.
SIGSEGV	Atminties adresavimo (“segment violation”) klaida.
SIGSYS	Neatstatoma klaida sisteminiame kreipinyje.
SIGPIPE	Rašymas į “vamzdį” neturint proceso – skaitytojo.
SIGALARM	Siunčiamas procesui iš branduolio kai baigiasi laiko limitas.
SIGTERM	Siunčiamas vartotojo proceso, kad pabaigtų kitą procesą.
SIGUSR1,2	Vartotojo apibrėžiami signalai, siunčiami iš taikomojo proceso.

Kadangi, kaip ja minėta, programuotojui suteikiamas signalų apdorojimo mechanizmas, taikomosios programos gali bandyti susidoroti su nepageidautinomis situacijomis. Signalams siekti naudojamas sisteminis kreipinys ***signal***, kuris turi du parametrus: stebimo signalo vardą ir pageidaujamų veiksmų aprašą. Tokiais veiksmais gali būti:

- signalo ignoravimas; - specialios situacijos apdorojimo funkcijos vykdymas;

- standartinės apdorojimo funkcijos vykdymas (paprastai, proceso nutraukimas).

Pavyzdžiui, kreipinys:

signal(SIGFPE, fpe_apdorojimas)

kur *fpe_apdorojimas* – vartotojo klaidos apdorojimo funkcija, kuri bus paleista vykdyti aptikus signalą SIGFPE.

Vienintelis nesiekiamas signalas - SIGKILL.

Signalų siuntimas.

Signalai siunčiami naudojant sisteminį kreipinį **kill**, kurio pavadinimas nusako kokių veiksmų bus imamasi pagal nutylėjimą ☺:

kill(pid, sig)

kur *pid* – proceso, signalo gavėjo, identifikatorius; *sig* – siunčiamas signalas, pvz.,

kill(457, SIGUSR2).

- **Vamzdžiai (pipes):** vieno proceso išėjimas nukreipiamas į kito proceso įėjimą.

Darbui su vamzdžiais dažniausiai naudojamos *Shell* komandos, kuriose "vamzdžio" operacija vaizduojama simboliu "**|**":

$ who | wc –l

Tokių komandų vykdymas reiškia laikinojo tarpinio failo kūrimą. Pirmas procesas rašys į šį failą, o antrasis skaitys iš jo. UNIX programuotojai gali ir patys sukurti toki "pipe" failą ir naudotis standartinėmis *read* ir *write* procedūromis. "Pipe" failas kuriamas sisteminiu kreipiniu **pipe**, kuris gražino du failo vardus (deskriptorius), rašymui ir skaitymui.

Paprastai sistema organizuoja "pipe" failą kaip eilę (t.y., FIFO struktūrą) ir taiko apribojimus šios eilės dydžiui (tipiškai 5120 bytes). Iš tuščios eilės negalima skaityti, į pilną eilę negalima rašyti.

Failų deskriptoriai gali būti paveldėti, tad "pipe" mechanizmas veikia tarp proceso "tėvo" ir visų jo "vaikų".

Problema: vamzdžiai egzistuoja tol, kol egzistuoja juos inicijuojantys procesai. Duomenys neperskaityti iki tokio proceso veikimo pabaigos bus prarasti. Dalinai problema sprendžia *vardiniai vamzdžiai* (named pipes).

- **Vardiniai vamzdžiai**

Vardiniai vamzdžiai (taip pat vadinami FIFO) – tai vamzdžiai, kurie naudoja pastovų UNIX sistemos failą. Jų sukūrimui naudojama *mknode* komandų procesoriaus komanda arba sisteminis kreipinys:
$ /etc/mknod npipe p
kur parametras "p" ir nusako, kad reikalingas "pipe" failas. Sukurtas failas nieko nesiskiria nuo paprasto failo, t.y., gali būti naudojimas daugelio kitų tarpusavyje nesusijusių procesų.

- **Pranešimų siuntimas.**

Pranešimų siuntimas panašus į vamzdžių, t.y., FIFO mechanizmą, nes ir čia procesai apsikeičia simbolių sekomis naudojant pranešimų eiles (*message queues*). Tokios eilės kuriamos sisteminiu kreipiniu *msgget* ir, karta sukurtos, tampa prieinamos ir kitiems procesams per *msgsnd* ir *msgrcv* sisteminius kreipinius. Kiekviena eilė pasižymi numeriu (sveikojo tipo kintamuoju), kuris tarnauja jos vardu. Patys pranešimai – tai struktūros (įrašai) iš dviejų dalių: teksto ir teksto tipo (long integer). Funkcija *msgrcv* leidžia atsižvelgti į pranešimo tipą, pvz., perskaityi pranešimus tipo (t.y., pvz., prioriteto) didėjimo tvarka.

Pranešimų eilės gali būti valdomos ar naikinamos naudojant sisteminį kreipinį *msgctl*.

- **Bendro naudojimo atmintis.**

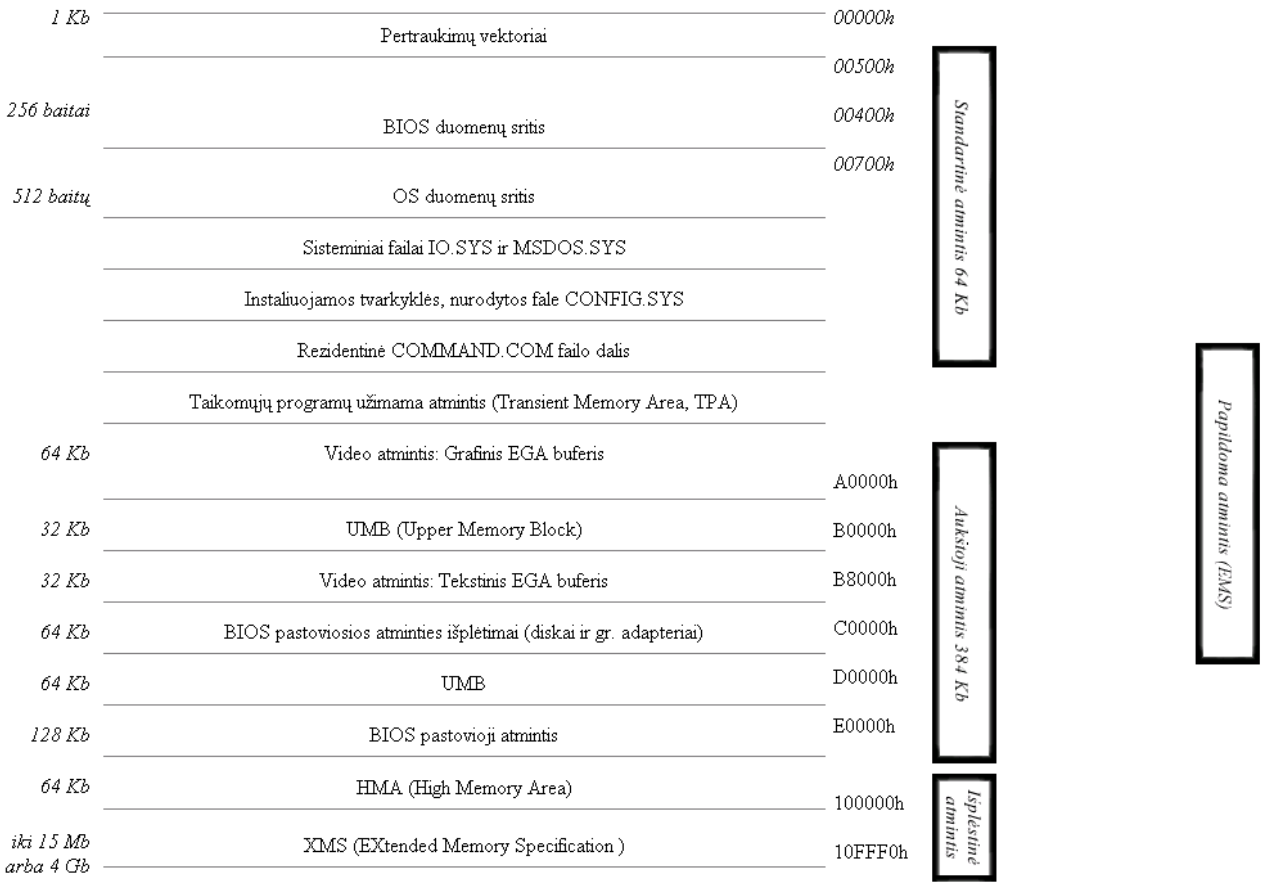
Tam tikra atminties sritis UNIX sistemoje gali būti skirta bendram procesų naudojimui. Tokia galimybė įgyvendinama sisteminiais kreipiniais *shmget, shmat, shmdt, shmctl*. Funkcija *shmget* veikia analogiškai *msgget*: tam tikrą atminties sritis bus išskirta ir atitinkamo segmento identifikatorius bus gražintas. Kiti procesai gali naudotis *shmget* šiam segmento identifikatoriui gauti.

Norint naudotis šia atmintimi procesas turi kreipiniu *shmat* prijungti bendrai naudojamą segmentą prie kitų jam skirtų atminties segmentų. Gavęs segmento adresą, procesas galės tiesiogiai kreiptis į reikiamą vietą. Funkcija *shmdt* atjungia (*detach*) bendrai naudojamą segmentą nuo proceso. Bendrai naudojamos atminties valdymą ir panaikinimą atlieka funkcija *shmctl*.

7. Atminties valdymas

7.1. Atminties valdymas MS-DOS sistemoje

Tipinis atminties pasiskirstymas ***(IBM PC, 80x86 procesorius)***



7.2. Pagrindiniai atminties valdymo uždaviniai

Pagrindiniai atminties valdymo uždaviniai yra:

- vesti atminties apskaitą; išskirti atminties vietas keliems procesams, vykdomiems vienu metu;
- užtikrinti pakankamą procesų vykdymo greitį;
- apsaugoti vykdomuosius procesus ir OS;
- leisti procesams bendrai naudotis atminties sritimis;
- suteikti programuotojui adresavimo ir atminties valdymo priemones.

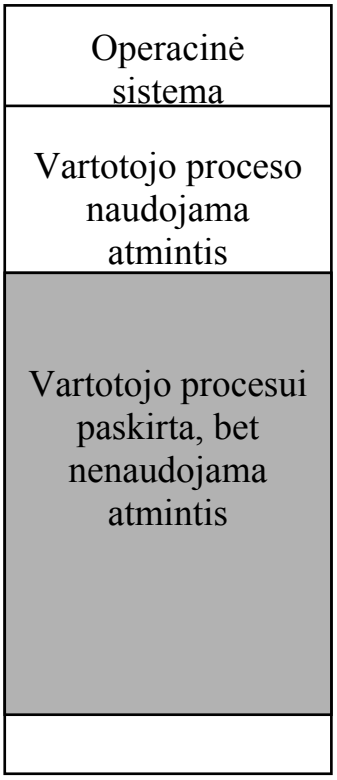
Fiziniai adresai – Loginiai adresai – Virtualiniai adresai

Procesų pakrovimas (process loading) ir apsikeitimas (swapping)

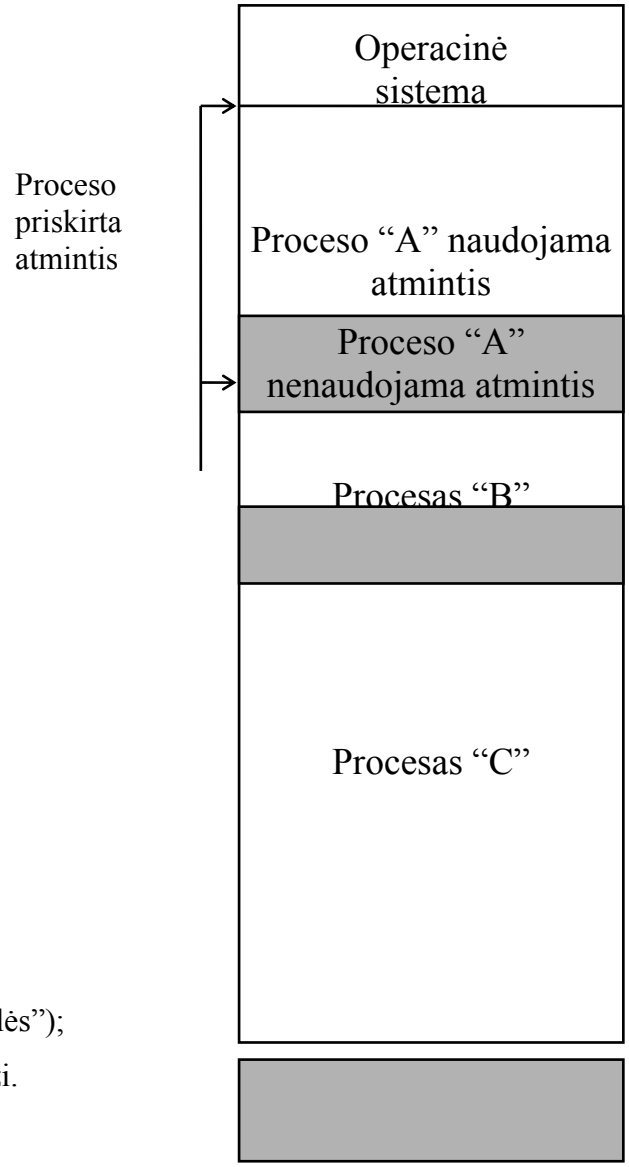
Perdavimo (transfer) laikas, priėjimo (access) laikas

Atminties skirstymo metodai

Vieno proceso (srities) skirstymas



Fiksuoto padalijimo metodas

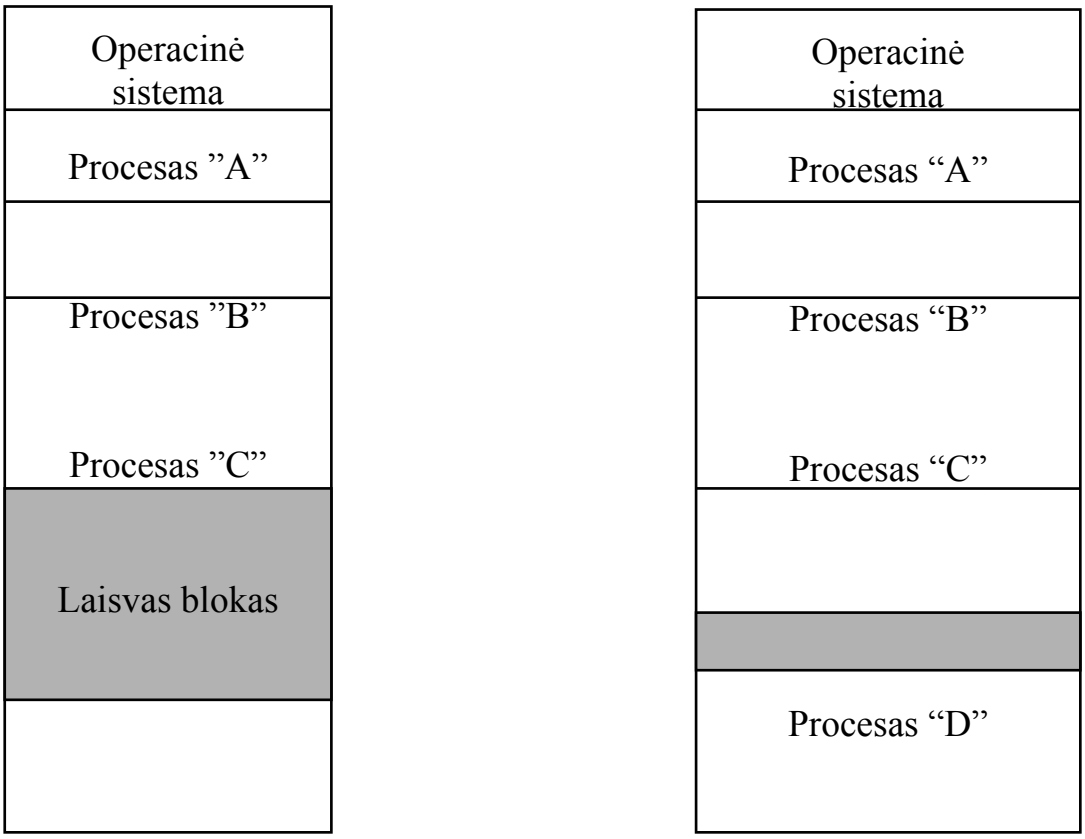


Trūkumai:

- vidinis (internal) fragmentavimas ("skylės");
- fiksuotų blokų dydžiai gali būti per maži.

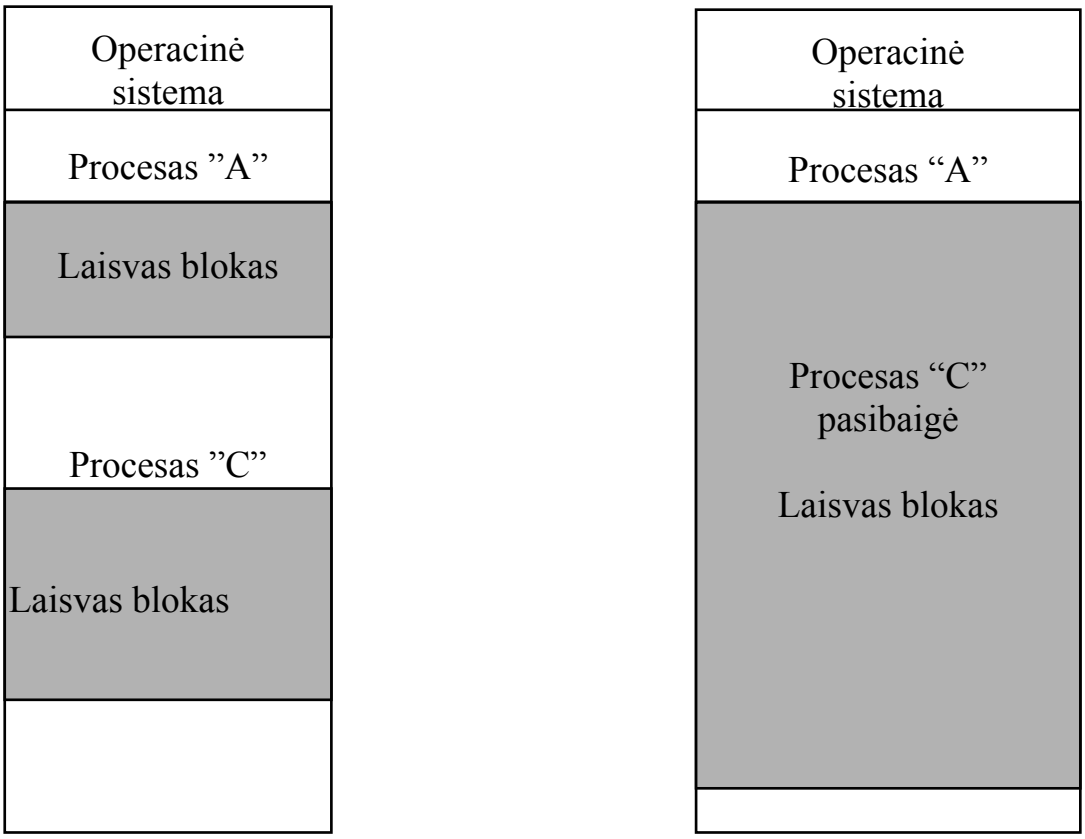
7.3. Atminties valdymo metodai

Kintamo padalijimo metodas



- išorinis (*external*) fragmentavimas;
- laisvų blokų susivienijimas (*coalescing*);

Fragmentavimas ir susivienijimas



7.4. Atminties skirstymo strategijos

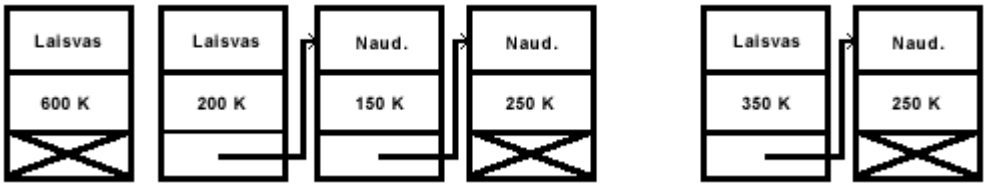
Atminties skirstymo strategijos

- geriausio atitikmens (*best fit policy*);
- pirmojo atitikmens (*first fit policy*);
- blogiausio atitikmens (*worst fit policy*);

Kintamo padalijimo metodas su suspaudimu (*compaction*)

Suspaudimas vykdomas: kai procesas pasibaigė; fiksuotais intervalais; vartotojui nusprendus; kai procesas negali būti pakrautas dėl fragmetavimo.

Blokų valdymo realizavimas, pvz., rišlus sąrašas:



Puslapiais skirstoma atmintis (*paging*)

Atminties skirstymo vienetas - fiksuoto dydžio puslapis.

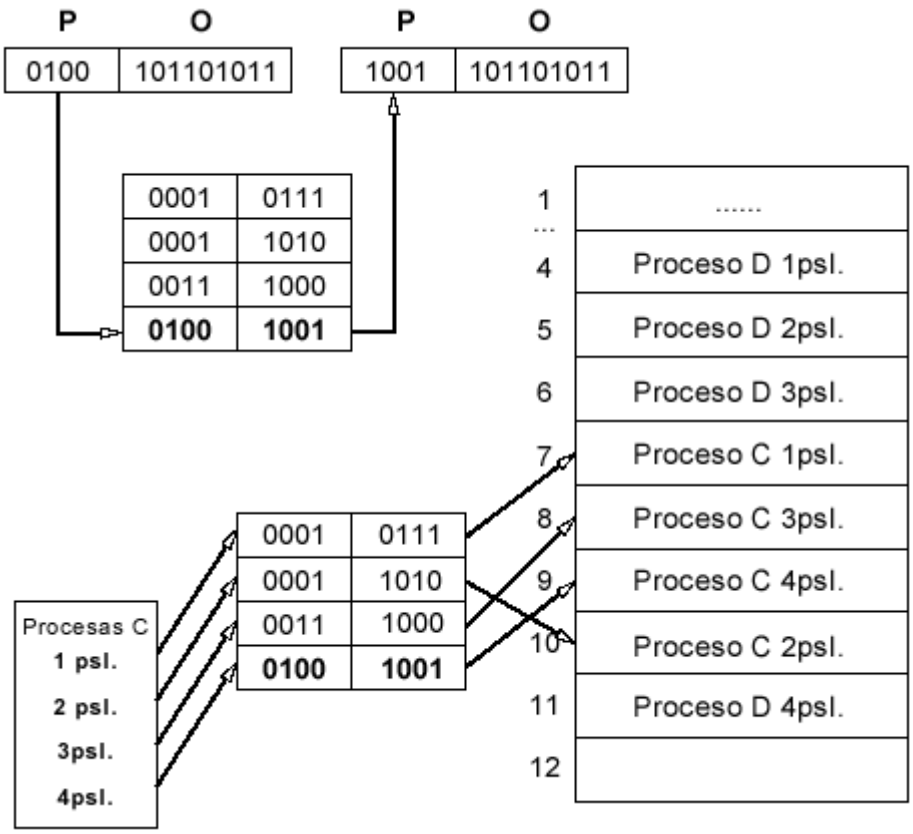
Procesų (loginiai) puslapiai vs atminties (fiziniai) puslapiai (blokai)

Procesas A 1 psl. 2 psl. 3 psl.	*Proceso A 1 psl.*	**1**	*Proceso A 1 psl.*
	Proceso A 2 psl.	**2**	*Proceso A 2 psl.*
	Proceso A 3 psl.	**3**	*Proceso A 3 psl.*
Procesas B 1 psl. 2 psl. 3 psl.	*Proceso B 1 psl.*	**4**	*Proceso D 1 psl.*
	Proceso B 2 psl.	**5**	*Proceso D 2 psl.*
	Proceso B 3 psl.	**6**	*Proceso D 3 psl.*
Procesas C 1 psl. 2 psl. 3 psl. 4 psl.	*Proceso C 1 psl.*	**7**	*Proceso C 1 psl.*
	Proceso C 3 psl.	**8**	*Proceso C 3 psl.*
	Proceso C 4 psl.	**9**	*Proceso C 4 psl.*
	Proceso C 2 psl.	**10**	*Proceso C 2 psl.*
Procesas D 1 psl. 2 psl. 3 psl. 4 psl.		**11**	*Proceso D 4 psl.*

7.5. Puslapiais skirstomos atminties valdymas

Adresas susideda iš dviejų dalių: puslapio numerio ir poslinkio puslapyje.

Puslapių lentelė konvertuoja loginio puslapio numerį į fizinį.

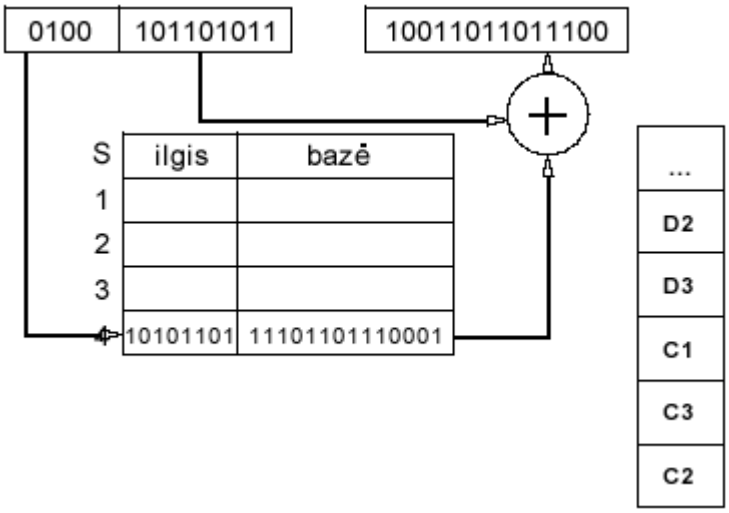


7.6. Segmentais skirstoma atmintis (*segmentation*)

Atminties skirstymo vienetas - kintamojo dydžio segmentas. Analogija su kintamo padalijimo metodu, bet procesas gali būti suskaidytas į segmentus.

loginis adresas *fizinis adresas*

segm. poslinkis bazė + poslinkis



• Segmentavimas atspindi loginę programos struktūrą ir todėl minimizuoja procesų pakrovimą. Leidžia procesams bendrai naudotis atmintimi.
• Puslapiavimas leidžia išvengti fragmentavimo; aiškus programuotojui.

7.7. Virtualioji atmintis

Prielaidos ir savybės:

• Procesas gali būti pakrautas į atmintį dalimis (puslapiais arba segmentais)

• Atminties adresai gali būti apskaičiuojami (transliuojami) dinamiškai

• Didesnis skaičius procesų gali būti vykdomas vienu metu

• Kiekvienas procesas gali turėti daugiau (virtualios) atminties nei realiai yra

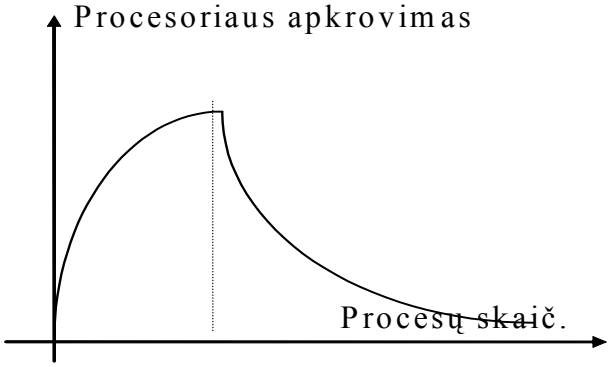
Virtualios atminties realizacijos mechanizmai

Puslapiais organizuojama virtualioji atmintis.

Vykdymo pradžioje pakrovėjas talpina į atmintį vieną puslapį. Toliau generuojami puslapių pareikalavimo pertraukimai. Formuojamas proceso rezidentinis rinkinys. Kai realioje atmintyje nebėra nevieno laisvo puslapio, OS vykdo puslapių apsikeitimą tarp pagrindinės ir diskinės atminties.

Kiekvieno proceso adresų transliavimas vyksta naudojant atskiras puslapių lenteles. Specialus puslapių lentelės registras saugo nuorodą į aktyvaus proceso lentelės adresą atmintyje. Keičiantis procesui keičiasi jo kontekstas (t.y. registrų turinys). Adresų transliavimą vykdo elektroninės (procesoriaus) schemos.

Be adresų, i-joje lentelės eilutėje saugojami i-jo puslapio kreipimosi požymių bitai: p, m ir kt. Tad, eilutės dydis - keletas baitų; puslapių skaičius ≈ 1 mln. Todėl, pačios puslapių lentelės saugojamos virtualioje atmintyje.



Sistemos apkrovimas. Nuorodų lokalizavimas. Proceso darbinis rinkinys. Darbinio rinkinio langas. Neproduktyvus apsikeitimas (thrashing).
Optimalus aktyvių procesų skaičius.

Puslapių apsikeitimo strategijos

• *LRU* - puslapis, mažiausiai naudotas pastaruoju metu (least recently used)

• *NRU* - nenaudotas pastaruoju metu

• *FIFO* - eilės metodas: keičiamas puslapis, kuris ilgiausiai buvo atmintyje

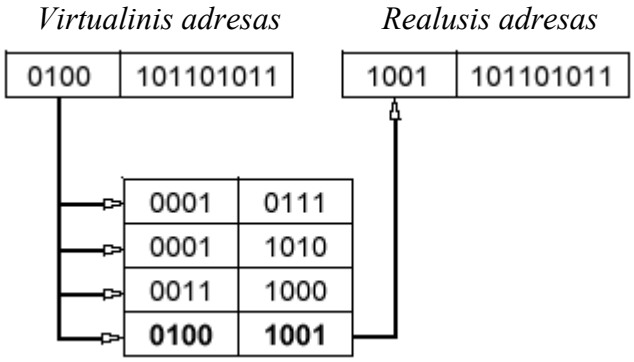
• *Clock* arba *second chance* - eilė kaip žiedinis sąrašas su panaudojimo bitu

• *NWPF* - išankstinis laisvų atminties blokų paruošimas puslapių pakrovimui

Elektroniniai virtualios atminties realizacijos mechanizmai

TLB - translation lockaside buffer- atskiras transliavimo buferis.

Associative mapping - asociatyvus atvaizdavimas.



Asociatyvus transliavimo buferis greitoje procesoriaus atmintyje

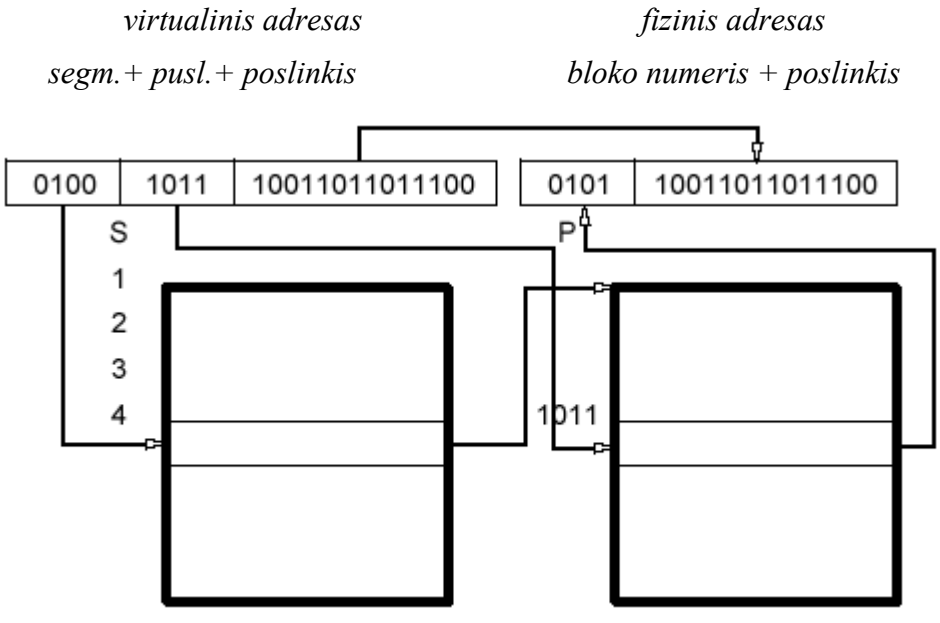
Segmentais skirstoma virtualioji atmintis

- palengvina dinaminės atminties panaudojimą;
- leidžia bendrai naudotis kodu ir duomenimis;
- geresne adresų lokalizacija, nes segmentais atvaizduota proceso struktūra.

Segmentų lentelės eilutė - *segmento deskriptorius.* Jo tipinis turinys:

bazinis segmento adresas; segmento max dydis; segmento požymių bitai, t.y. būvimo realioje atmintyje (y,n), panaudojimo (u), apsaugos (r,w,e).

Kombinuotas puslapių ir segmentų metodas (MS-Windows, OS/2)



Proceso segmentų lentelė Segmento puslapių lentelė

Atminties apsaugos ir bendro naudojimo aspektai

Violation - proceso bandymas adresuoti jam nepriklausomą atmintį Adresavimo klaidų neatpažįsta transliatoriai; naudojama dinaminė atmintis

Ribiniai registrai (limit registers) saugo viršutinius skiriamų blokų adresus.

Puslapiavimo sistemose specialus registras saugo aktyvaus proceso maksimalų puslapio numerį. Poslinkis negali viršyti puslapio dydžio. Puslapių turinys paprastai nežinomas, tad retai bendrai naudojamas.

Segmentavimas atspindi loginę programos struktūrą, segmentų turinys - tai programuotojo apibrėžtos procedūros ir duomenys. Todėl priėjimo kontrolė ir bendras segmentų naudojimas nesunkiai įgyvendinami naudojant *segmento deskriptoriaus atributus.*

7.8. Personalinio kompiuterio OS atminties modeliai

MS-DOS:

• Naudojamas segmentavimas, segmentų registrai gali būti keičiami vykdymo metu. Procesas gali turėti daug kodo ir/arba duomenų segmentų.

• Naujos versijos leidžia turėti atmintyje kelis procesus, tik vienas iš kurių yra aktyvus. Proceso pakrovimui į laisvą bloką naudojamas pirmojo atitikmens metodas.

• Atminties modelių apibrėžimai:

Modelis	Kodo segmentų skaičius	Duomenų segmentų skaičius	Savybės
Tiny	1		Viskas - kodas, duomenys, krūva ir stekas - užima 64K. (*.com* failai)
Small	1 nuorodos “near”	1 near	Statiniai duomenys, krūva ir stekas kartų užima 64K.
Compact	1 near	>1 far	Kodas, stat. duomenys ir stekas - po 64K; Din. duomenis < 1Mb.
Medium	>1 far	1 near	Kodas <1Mb, statiniai duomenys krūva ir stekas kartų užima 64K.
Large	>1 far	>1 far	Kodas <1Mb, krūva < 1Mb; statiniai duomenys - 64K.
Huge	>1 far	>1 far	Kodas <1Mb, krūva < 1Mb; statiniai duomenys > 64K.

• Du adresavimo tipai: “artimas” *near* ir “tolimas” *far*. *Near* - procesoriaus instrukcijose naudojamas tik poslinkis; *far* - instrukcijose naudojamas tiek segmento adresas, tiek poslinkis. *Near* adresavimui reikia mažesnio procesoriaus taktų skaičiaus; tokios komandos vykdomos greičiau.

• Du objektinių programų formatai:

.*COM* - naudoja tik vieną segmentą tiek kodui, tiek duomenims (t.y. 64K). Čia segmentų registrai CS, DS, SS, ir ES įgauna vienodą reikšmę.

.*EXE* - šios programos gali naudoti visus segmentų registrus (iki 6) ir, atitinkamai, užimti daug atminties segmentų tiek kodui, tiek duomenims.

MS-WINDOWS 3.x:

• Skirtingai nuo OS/2 šios Windows versijos vis dar naudoja 16 bitų atminties adresavimą, t.y. nesugeba pasinaudoti 80386 ir naujų procesorių 32 bitų adresavimo galimybėmis. Tai savotiškas mokestis už suderinamumą su senais Intel procesorių modeliais.

• Trys Windows 3.x darbo režimai:

Realusis režimas (*real mode*) skirtas Windows 3.0 versijai, 8086/8088 procesoriams ir yra analogiškas MS-DOS režimui. Vartotojui prieinama tik iki 640 Kb operatyviosios atminties.

Standartinis režimas (*standard mode*) skirtas versijoms 3.0 ir 3.1 bei 80286 procesoriui (dar vadinamas apsauginiu 286 režimu). Reikalauja mažiausiai 1Mb (v. 3.0) ir 2Mb (v. 3.1) op. atminties bei 6.5-9 Mb diskinės atminties. Leidžia naudotis visa išplėstine atmintimi (iki 16Mb) ir persijungti iš vienos programos į kitą.

Išplėstinis režimas (*enhanced mode*) skirtas Windows sistemoms vykdomoms 80386 ar naujesniuose procesoriuose (dar vadinamas apsauginiu 386 režimu). Realizuoja virtualinės atminties (mašinos) režimą, leidžia vykdyti daug uždavinių vienu metu (tame tarpe ir keletą MS-DOS programų).

Naudojamas segmentavimas, leidžiantis naudoti tą patį kodo ir resursų segmentą kelioms tos pačios programos kopijoms. Kodo ir resursų segmentai gali būti pakrauti pagal pareikalavimą.

Atminties blokai gali būti perkeliami vykdymo metu; tokiais blokais gali būti kodo, duomenų ir resursų segmentai. Jie vadinami "*movable*".

Segmentai gali būti ir griežtai fiksuoti "fixed"; juose talpinamos MS-DOS programos vykdomos Windows sistemoje.

Segmentus galima pažymėti kaip iškraunamus - "*discardable*". Segmentų apsikeitimui naudojamas modifikuotas FIFO metodas.

MS-WINDOWS ' 95:

• Programuotojo taikymų interfeisai: Win16 API, Win32 ir Win32s API. Adreso transformavimas iš vieno formato į kitą -"thrunking".

• Windows' 95 virtualinės atminties pasiskirstymas:

<table>
<tr><td rowspan="7">FFFFFFFF

BFFFFFFF

07FFFFFF

003FFFFF

000FFFFF

00003FFF

0</td><td>Operacinė sistema</td><td>Kernel, GDI, VxDs, privilegijuoti sisteminiai
DLL tik MS-DOS programos 3</td><td>Sisteminė sritis</td><td>4Gb

3Gb</td></tr>
<tr><td rowspan="2"></td><td>nesisteminiai DLL (pvz., COMDLG32.DLL)
ir kiti bendrai naudojami objektai</td><td>Bendrai naudojama sritis</td><td>2Gb</td></tr>
<tr><td>Programos</td><td>Win32 (ir Win16) programos</td><td rowspan="4">Taikymų Sritis</td><td>4Mb</td></tr>
<tr><td></td><td>tik Win16 programos</td><td>1Mb</td></tr>
<tr><td></td><td>2-bitų taikymų neadresuojama atmintis</td><td>16K</td></tr>
<tr><td></td><td>32-bitų taikymų neadresuojama atmintis</td><td>0</td></tr>
</table>

• Virtualios atminties valdymas. Skirtingai nuo Windows 3.x naudojamas kintamo dydžio *swap* failas, bet atsiranda failo fragmentavimo problema, kuri sprendžiama pirmojo atitikmens metodu. Tiesioginiai kreipiniai į OS.
• Apsaugotos sritys. Nauji dinaminės atminties išskyrimo būdai.
• Windows NT: sisteminė adresų sritis yra apsaugota nuo taikymų.
• Atvaizduojamos atminties failai: rekomenduojami bendro naudojimo atminties išskyrimui didelėms rezidentinėms struktūroms. Tarp 2Gb ir 3Gb.
• Taikymo virtualinės atminties rezervavimas: *VirtualAlloc()*.
• Taikymo lokali dinaminė atmintis: *HeapCreate()*. Fizinė atmintis neskiriama.

UNIX :

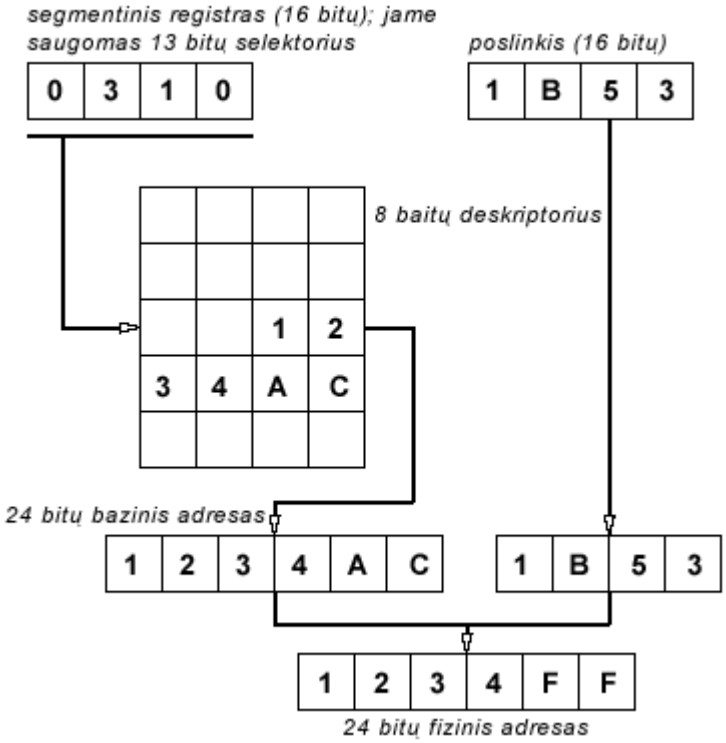
• Daug įvairių versijų. Suderinamumui užtikrinti naudojamas standartinis kiekvieno proceso skirstymas į 3 segmentus: teksto, duomenų ir steko. Teksto segmente yra vykdomasis proceso kodas, duomenų - inicializuoti ir neinicializuoti duomenys. Steko segmente gali būti procedūrų ir funkcijų parametrai bei lokaliniai kintamieji, kiti dinaminiai duomenys. Segmentai gali būti bendro naudojimo, pvz., bendra duomenų sritis, naudojama informacijos tarp procesų apsikeitimui.

• Šiuolaikinės UNIX (LINUX) versijos atminties valdymui naudoja virtualinį puslapiavimą. Tipinis puslapio dydis - 4Kb. Periodiškas sisteminis *daemon* procesas palaiko laisvų atminties blokų skaičių tarp *minframe* ir *maxframe* reikšmių. Kaip ir ankstesnės versijos dabartinės sistemos gali pašalinti iš atminties ištiesinį procesą; tam naudojamas LRU metodas...

7.9. Virtualinės atminties palaikymas Intel mikroprocesoriuose

80286 apsauginis režimas *(protected mode).*

Lokali ir globali deskriptorių lentelės *(LDT, GDT).* Registrai LDTR ir GDTR ; jų turinio keitimas - privilegijuota operacija (ją atlieka tik OS).



Apsauga: procesas gali naudotis tik tais segmentais, kurių deskriptoriai yra LDT. Bendras naudojimas: OS naudoja GDT tam, kad leistų procesams bendrai naudotis atminties blokais.

80386 (486, 586) apsauginis režimas. Lentelių ir deskriptorių formatai.

Realiam režime adresavimas analogiškas 8086, aukštieji 16 bitų ignoruojami. Apsauginiame režime procesorius (kaip ir 80286) naudoja segmento registro turinį kaip indeksą deskriptorių lentelėje. Toliau deskriptorius naudojamas puslapių adresavimui ir po to fizinio adreso apskaičiavimui.

Deskriptorius:

Bazinio adreso bitai 0..15	Segmento ribos bitai 0..15

Bazinio adreso 24..31 bitai	G	B D	Ribos bitai 16..19	D P	P L	S	Tipas	A	Bazinio adreso bitai 16..23

32 bitų adresas, kuris nurodo reikiamo segmento pradžią, formuojamas iš 4 bazinio adreso laukų; adresą segmente nusako poslinkis. *Bazinio adreso bitai 24..31* nenaudojami, jei vykdoma programa, skirta 286 procesoriui (24 bitų adresas). Segmento dydį nusako 20 bitų segmento ribos adresas ir *granuliacijos bitas G*, turintys reikšmes: 0 - "baitais", 1 - "puslapiais".

B=1, kai duomenų segmentas užima daugiau 64K; *D=1*, jei segmente yra 386 kodas ir 0 jei 286. P - segmento buvimo realioje atmintyje požymis.

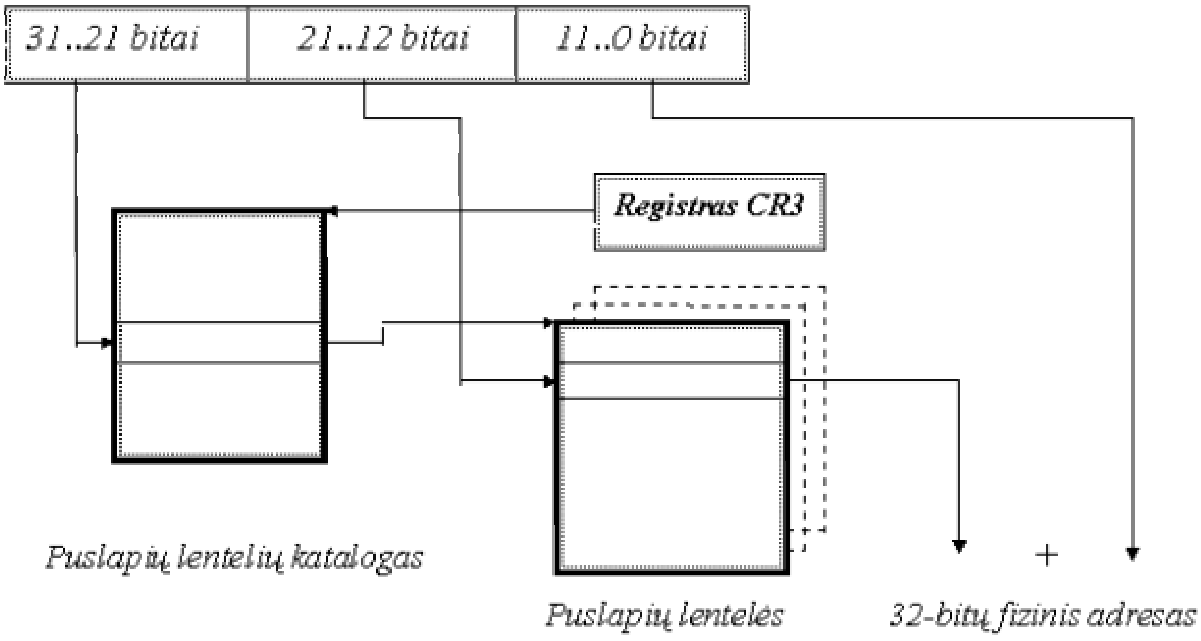
DPL - segmento privilegijos lygmuo (Windows naudoja tik 0 ir 3)

S=1, jei aprašomas atminties segmentas (gali būti pertraukimų ir kt.)

Tipas - atminties segmento tipas, t.y., kodas, duomenys, r, w, a ir pan.

A=1, jei kuris nors procesas jau turėjo nuorodą į šį segmentą (naudojimas).

Virtualinio adreso interpretavimas (puslapių adresavimas)



Lentelių elementų formatai

Lentelės numeris arba Puslapio adresas	bitai 31..12		D	A			U	R S	P W	

D (dirty) - puslapio turinio pasikeitimo požymis.
A (access) - bet kuri nuoroda į puslapį daro A=1. Puslapių šalinimui iš atminties Windows'95 naudoja LRU algoritmą.
U/S (user/supervisor) - U=0, kai puslapis skirtas supervizoriui (OS). Jei vartotojo programa kreipsis į tokį puslapį, OS nutrauks jos veikimą ir pateiks pranešimą "*General Protection Fault"*.
R/W (read/write) -paprastai programos kodas žymimas "tik skaitymui" ("0").
P (present) - P=1, kai puslapis arba puslapių lentelė yra atmintyje.

8. Failų valdymas

Failas – tai "kažkokia" informacija saugojama išorinėje atmintyje. Failo vartotojas paprastai susiduria su loginiu informacijos vienetu, tuo tarpu fizinėmis operacijomis rūpinasi OS. Toliau apžvelgiami tiek OS servisai, leidžiantys vartotojui dirbti su failais, tiek mechanizmai, kurie naudojami OS viduje atliekant tokias paslaugas.

Failų tipai. Ar turi OS žinoti apie failų struktūrą?
Kai kurios sistemos žino apie failų struktūrą (failų tipus), palaiko skirtingus organizacijos būdus, pvz., indeksinius, indeksinius – nuoseklius, kintamo įrašų ilgio ir pan. Jose taikomosios programos gali naudotis tokiais iš anksto paruoštais servisais. Be to, atsiranda galimybė išvengti neteisingų operacijų panaudojimo, pvz., binarinio failo apdorojimo teksto redaktoriumi. Tačiau labai išaugo tokių sistemų sudėtingumas ir dydis. Taip pat gali atsirasti poreikis kitai, nenumatytai failo organizacijai.

UNIX ir MS-DOS sistemos atsiriboja nuo failų struktūros: failai įsivaizduojami kaip nestruktūrizuotos baitų sekos. Taikomosios programos turi pačios pasirūpinti joms reikalingą failų struktūra, pvz., sudaryti failą iš įrašų, nustatyti įrašų ilgį ir pan. Šios OS nenustato nekorektiškų (failo turinio atžvilgiu) operacijų.

UNIX skiria tokius "failų tipus":

normalus (regular)	įprasti failai, t.y., programos ir duomenys.
katalogas (directory)	failas turintis nuorodas į kitus failus.
simbolinis/blokinis (char/block special)	specialūs failai - nuorodos į įtaisus.
vamzdis (pipe)	eilės tipo buferis.

Speciali programa ***file*** naudojama failo turinio atpažinimui ir leidžia nustatyti vykdomųjų programų failus, programavimo kalbų kodus ir pan.

UNIX ir MS-DOS ir Windows sistemos grupuoja failus į katalogus (aplankus). Faktiškai katalogai – tai specialaus pavidalo failai.

MS-DOS sistemoje failai gali turėti skirtingus **atributus** (kelis iš karto), kurie skirti administravimo tikslams ir beveik nenaudojami taikomosiomis programomis:

system	skiriamas sisteminiams failams;
archive	naudojamas failų rezervinei kopijai pasidaryti;

hidden paprastai ignoruojamas sisteminėmis komandomis;

read-only failas negali būti pakeistas arba pašalintas.

8.1. Failų identifikacija

MS-DOS. Failo vardo ilgis – 12 simbolių, iš kurių 8 simboliai – pats vardas, vienas – taškas (atskyrėjas) ir dar 3 simboliai – vardo išplėtimas (turinio tipas). Vardo viduje naudojamos raides, skaičiai ir kai kurie specialūs simboliai ($ # &~@)).

UNIX. Skirtinga skirtingose versijose, bet ne tokia griežta kaip MS-DOS. Tipinis failo vardo ilgis – 14 simbolių, bet kai kuriose versijose (pvz., BSD, SVR4) iki 255 simbolių. Vardas nėra struktūrizuotas. Vardo viduje gali būti naudojami beveik visi simboliai, išskyrus / ir tarpo. Kai kurie simboliai turi tam tikras reikšmes komandų procesoriaus shell komandose, pvz.: * > ?.

Windows ir OS/2. Failo vardo ilgis gali būti iki 255 simbolių. Be įprastų simbolių vardo viduje gali būti naudojami tarpai ir kai kurie specialūs simboliai. Komandinėse eilutėse tokie vardai naudojami kabutėse.

Palaikomi ir MS-DOS vardai (tiksliau MS-DOS stiliaus aliasai), kurie generuojami iš ilgų vardų naudojant pirmus 8 ilgo vardo simbolius. Jei gaunamas vardas yra neunikalus, naudojami pirmieji 6 simboliai, tilda ~ ir numeris 1. Jei ir toks vardas yra neunikalus, tai skaičius yra didinamas vienetu: 2, 3 ir t.t.

8.1.1. Katalogai (direktorijos, aplankai)

Katalogizavimas – tai loginis failų grupavimas. Pirmosiose sistemose – tik vieno lygmens katalogai. Vėliau įvesta katalogų hierarchija: išskiriamas pagrindinis (root - šakninis) katalogas (žymimas / UNIX sistemoje ir \ MS-DOS) į kurį įeina pirmojo lygmens katalogai, į kuriuos įeina antros eilės katalogai ir t.t. Taip sukuriama medžio tipo informacijos struktūra. *Pastaba. Žodis "katalogas" suprantamas dvejopai: tai ir failų grupė ir specialus failas, kuriame ši grupė yra aprašyta.*

Kataloge sugojama tokia informacija apie į jį įeinančius failus: failo vardas, tipas, atributai, informacija apie failo lokaciją diske, priėjimo teisės, failo dydis baitais, sukūrimo ir/arba pakeitimo data, ir kt. Dar vienas katalogo sistemos privalumas – vienodų vardų failai gali būti saugojami skirtinguose kataloguose. Procesas paprastai dirba su failais, esančiais vadinamajame darbiniame kataloge (*current directory* arba *present working directory*).

Pagrindinės komandos darbui su katalogais:

cd Pakeisti katalogą (t.y.pereiti į kitą). Pvz., UNIX komanda ***cd /*** arba MS-DOS cd \ keičia darbinį į aukštesnio lygio katalogą.

mkdir Sudaryti katalogą. Pvz., komanda ***mkdir /user*** sudarys naują ***user*** katalogą darbiniame kataloge (MS-DOS – sutrumpinta ***md*** komanda).

rmdir Pašalinti katalogą. Pvz., komanda ***rmdir user*** pašalins ***user*** katalogą iš darbinio katalogo (MS-DOS – sutrumpinta ***rd*** komanda).

8.1.2. Keliai (*path*) ir jų apibrėžimas

Kelias į reikiamą katalogą gali būti nurodytas pradedant arba nuo pagrindinio (root) katalogo arba nuo darbinio (aktyvaus) katalogo; atitinkamai nurodomas arba pilnas kelio vardas arba santykinis vardas. Pvz., jei failas *failas1.dat* yra kataloge */cprogs/work/ failas1.dat* ir darbinis katalogas yra */cprogs,* tai pilnas kelio vardas būtų */cprogs/work/ failas1.dat,* o santykinis - *work/ failas1.dat* .

Kitas kelio nurodymo mechanizmas – paieškos kelio nurodymas komanda ***PATH***, pvz., MS-DOS sistemoje (paprastai, *autoexec.bat* faile) galima nurodyti:

PATH= \; \DOS; \USERS\JONAS\TEKSTAI

UNIX sistemoje kiekvienas vartotojas gali turėti savo kelių nurodymus. Vartotojo keliai paprastai nustatomi registravimo metu vykdant komandas iš failų *.profile* arba *.login* ir priskiriant čia apibrėžtus kelius kintamajam *PATH* (ar *path*). Tipinė komanda: ***PATH= /usr/staff/jonas/progs***

Pastaba. Kai keliai apibrėžiami naudojant *path* mechanizmą, yra pavojus įvykdyti ne tą failą, kurį iš tikrųjų norėjo vartotojas; pvz., jei komandinis failas pavadinimu *istrink.bat* yra tiek \DOS, tiek \USERS\JONAS kataloguose.

8.1.3. Aliasų vardai

UNIX leidžia naudoti aliasus, t.y., priskirti vienam fiziniam failui du skirtingus vardus, arba netgi tą patį vardą skirtinguose kataloguose. Jei failas turi būti prieinamas iš skirtingų katalogų, jis dedamas į vieną iš jų, o kituose daromos nuorodos (*link*) į jį. Tai galima padaryti sisteminiu kreipiniu *link()* arba shell komada ***ln:*** pvz., esant kataloge */cprogs/work/* galima sukurti vardą *myprog*

ln /usr/staff/jonas/progs myprog

kuris tarnaus kaip nuoroda iš */cprogs/work/* į programą kataloge */usr/staff/jonas/progs* .

8.1.4. Tomai (volumes)

Tomas – tai konkretus išorinio duomenų saugojimo įtaiso vienetas, diskelis, diskas, kompaktinis diskas ir pan. Prieš naudojant tomą reikia inicializuoti. MS-DOS sistemoje tai atliekama komanda *FORMAT*, UNIX sistemoje, paprastai, *mkfs (make file system)*. Šios komandos sukuria disko struktūrą ir atitinkamas failų aprašų lenteles.

Viename fiziniame tome gali būti keli loginiai tomai. Suskaidyti vieną fizinį diską į kelis loginius galima įvykdant žemio lygio pagalbinę skaidymo (*partitioning*) programą. Tai leidžia, pvz., turėti viename diske dvi skirtingas OS.

8.2 Sisteminiai failų valdymo servisai

Vartotojo operatyvios priemonės:

- failo kūrimas;
- failo šalinimas;
- failo kopijavimas;
- pervardinimas;
- failo turinio peržiūra;
- katalogo kūrimas;
- katalogo turinio peržiūra;
- tuščio katalogo šalinimas.

Programavimo priemonės:

- failo atidarymas;
- failo uždarymas;
- failo turinio skaitymas;
- rašymas į failą;
- pozicijos faile pasirinkimas (seek).

Programavimo priemonės aukšto lygio kalbose įgyvendinamos arba **bazinėse konstrukcijose** (pvz., INPUT – BASIC kalboje), arba **standartinėse kalbos procedūrose** (pvz., READ – PASCAL kalboje), arba atskirose bibliotekose (pvz., fread – C kalboje). Assembler kalboje darbui su failais

paprastai naudojami programiniai pertraukimai, pvz., MS-DOS sistemoje *int 21* – tai sisteminis kreipinys, leidžiantis atlikti aukščiau minėtas operacijas.

Darbas su katalogais parprastai tiesiogiai nepalaikomas aukšto lygio kalbose, bet daugelis programinių kompanijų pateikia papildomas standartines bibliotekas darbui su katalogais (ir failais) atlikti.

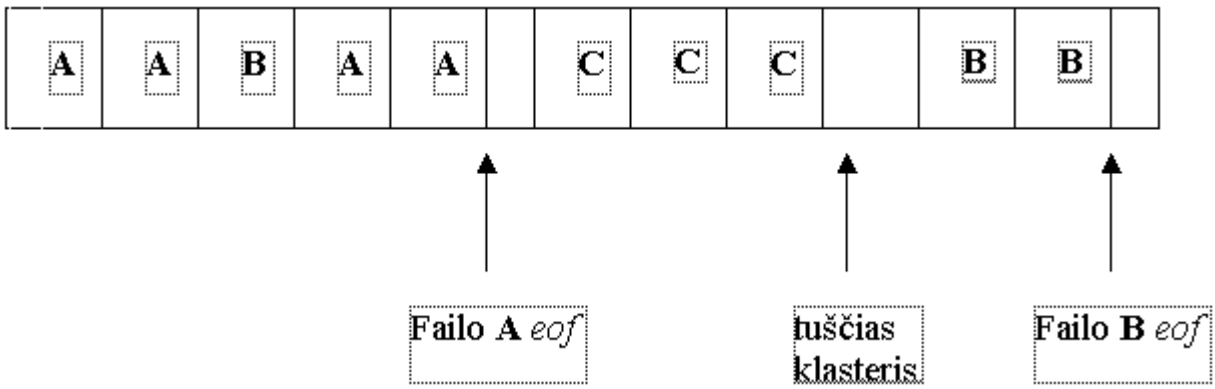
8.3. Failų valdymo metodai

Failų saugojimas ir vietos išskyrimas.

Ankstesnėse sistemose atsakingas už failo išdėstymą (pagal cilindrus, takelius, blokus) buvo vartotojas. Tad darbo efektyvumui padidinti failai buvo dėstomi iš eilės, nuosekliai, arba užimdavo tą patį cilindrą. Vartotojas turėjo pat apskaičiuoti kiekvieno failo dydį ir atitinkamai rezervuoti vietos diske. Tačiau nuoseklus (ar "vertikalus") failo dėstymas yra labai netaupus. Ši situacija yra analogiška atminties fragmentavimui, su visais panašiais trūkumais.

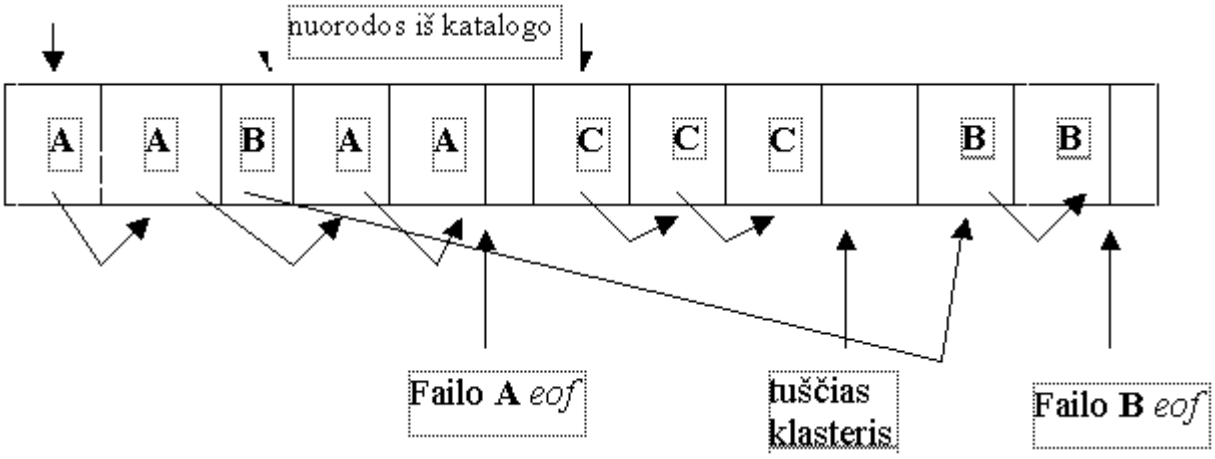
Šiuolaikinės sistemos išskiria failui vietą dinamiškai, iškilus konkrečios programos poreikiui. Čia vartotojas nieko nežino apie realią failo vietą diske. Paprastai (plg. su atminties puslapiavimu) reikalinga vieta išskiriama fiksuoto dydžio porcijomis – *allocation unit*. MS-DOS sistemoje tokia porcija vadinama **klasteriu** (*cluster*), kuris jungia fizinius disko sektorius į porcijas po 512, 1024, 2048 baitus. UNIX sistemoje išskyrimas vyksta sektoriais.

Kai failas yra ką tik sukurtas, jis susideda iš 1 klasterio, prie kurio vėliau jungiami kiti. Jei tai pirmas failas diske, jis bus išdėstytas nuosekliai, klasteris po klasterio, tačiau vėliau skirtingų failų klasteriai susimaišys:



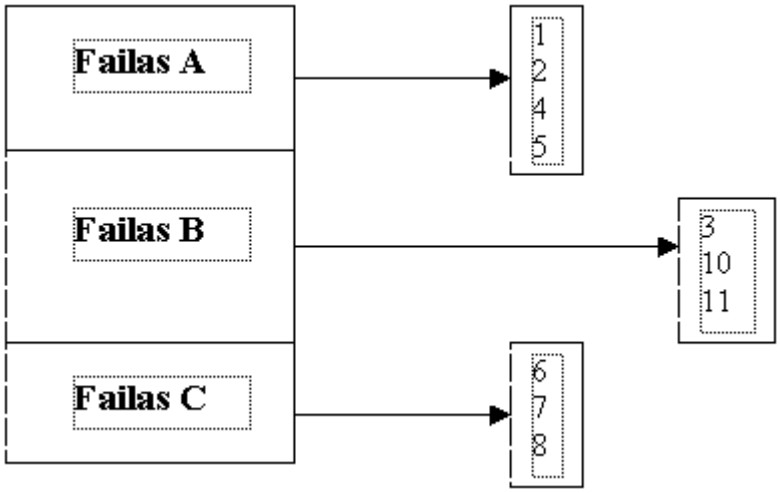
Failai išlaisvina klasterius, kai kinta jų dydis, arba jie naikinami. Tada tarp užimtų klasterių atsiranda tušti. OS turi sekti kurie klasteriai priklauso kokiam failui ir optimizuoti erdvės panaudojimą.

Pirmas metodas – klasterių sujungimas į grandinę (sąrašą), kurioje kiekviename klasteryje taip pat saugojamas kito (tolesnio) klasterio numeris.



Nuoroda į kitą klasterį užima vietą ir apsunkina failo skaitymą ir rašymą į failą. Bet blogiausia, kad failo skaitymas (rašymas) įmanomas tik nuosekliai per klasterių grandinę, o ne tiesiogiai. Tai vyksta labai lėtai ir gali būti nepriimtina daugeliui taikymų.

Antras metodas: failą sudarančių klasterių sąrašas, saugojamas kiekvienam failui.



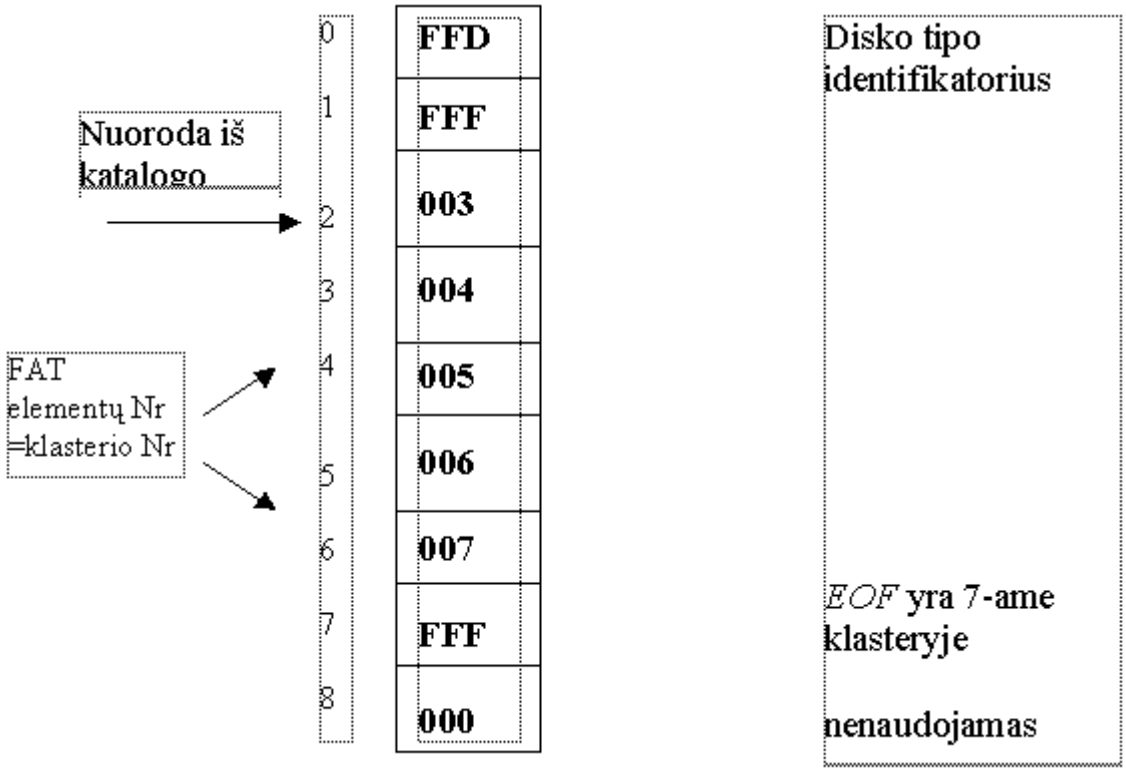
Šių metodų modifikacijos naudojamos atitinkamai MS-DOS ir UNIX sistemose.

8.4. MS-DOS failų sistema

Failų išdėstymas. Kai kurios sistemos naudoja tiesioginę duomenų klasterių grandinę, bet tai veikia lėtai ypač dideliems failams. MS-DOS sistemoje nuorodų lentelė atskirta nuo pačių duomenų klasterių.

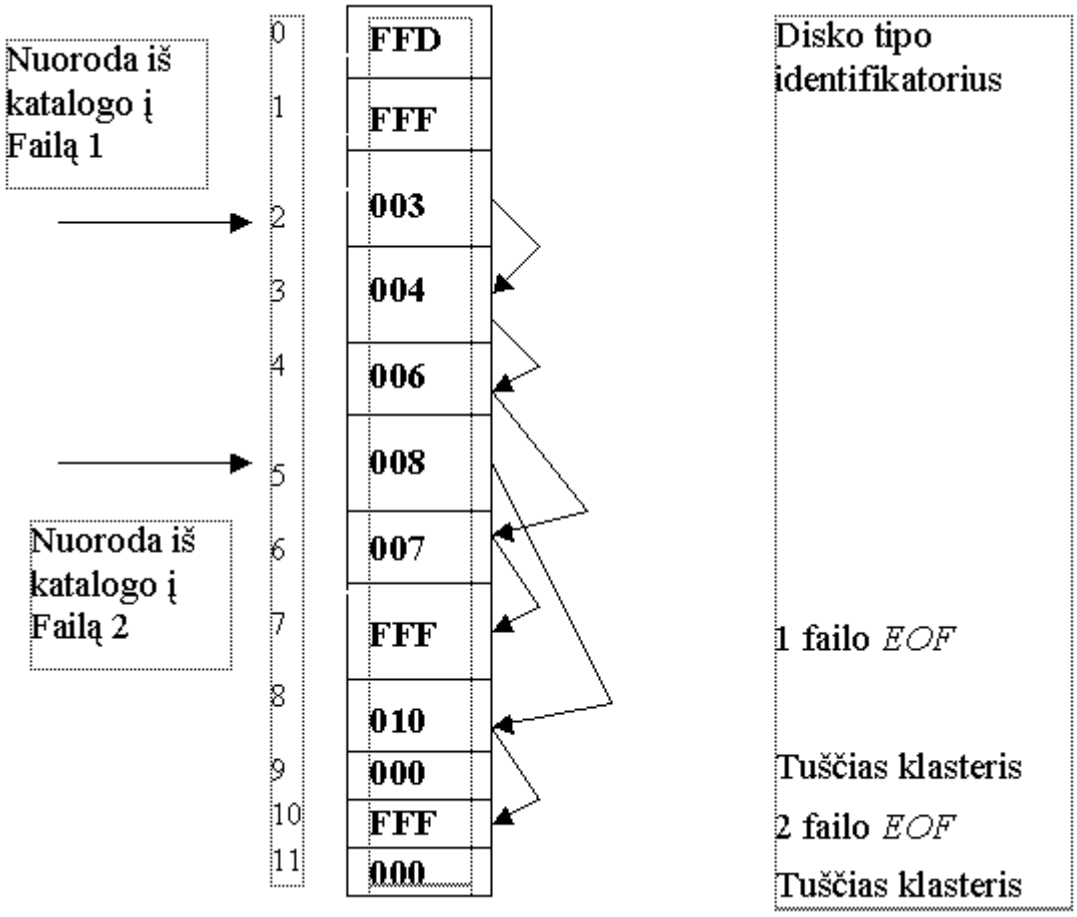
Tokia lentelė vadinasi FAT (*File Allocation Table)* ir susideda iš masyvo, kuriame saugojamos 12 bitų (diskeliui) arba 16 bitų (kietam diskui) reikšmės. Kiekvienas masyvo elementas atitinka vieną klasterį (išskyrus 2 pirmuosius). Šių elementų reikšmės – tai nuorodos į kitus lentelės elementus. Taigi vietoj sujungtų klasterių naudojamą nuorodų lentelė FAT.

Kai diskas tuščias, visi FAT elementai turi reikšmę 0 (tuščių klasterių požymis). Pirmam įrašytam failui bus sudaryta lentelė, kurioje klasterių nuorodos išdėstytos iš eilės:



Pirmas elementas identifikuoja disko tipą (šiuo atveju FFD reiškia – abipusis su 9 sektoriais / per takelį), o antrasis visada turi reikšmę FFF (ar FFFF kieto disko atveju). Grandinės pabaiga taip pat žymima FFF.

Tipinėje situacijoje (failų klasteriai sumaišyti) FAT struktūra atrodo taip:



Tiek FAT elementų skaičius, tiek jų dydis (bitų skaičius) priklauso nuo disko dydžio. Bet netgi maksimalus FAT dydis vis tiek leidžia saugoti ir apdoroti šią lentelę atmintyje ir nesikreipti į diską, todėl apdorojimo laikas yra patenkinamas.

Laisvos vietos kontrolė vykdoma priskiriant išlaisvinto klasterio elementui reikšmę (0)000. Tokia reikšmė leidžia suteikti klasterį naujam failui.

Tomo struktūra MS-DOS sistemoje.

Pakrovimo sektorius (*boot sector***)**. Susideda iš mašininės nukreipimo komandos *Jump*, kuri perduoda valdymą **pradiniam pakrovėjui.** Pastarasis suranda *IO.SYS* ir *MSDOS.SYS* failus, pakrauna juos į atmintį ir perduoda jiems valdymą. Pakrovimo sektoriuje taip pat įrašomas gamintojo vardas ir versija, bei failų išdėstymo detalės, pvz., sektoriai/klasteriai, FAT skaičius ir kt.

FAT – pagrindinė failų dėstymo lentelė.

Papildoma (-os) FAT – sistema naudoja dublikatines lenteles (1 ar daugiau) saugumui padidinti.

Šakninis katalogas – skiriasi nuo kitų fiksuotu dydžiu ir pozicija diske. Paprastai gali turėti nedaugiau 112 elementų.

Failų erdvė *(file space)* **–** likusi disko erdvė naudojama failų ir katalogų saugojimui.

Klasterio dydžio pasirinkimas. Priklausomai nuo disko tipo klasterio dydis gali būti: 512, 1024, 2048, 4096… Tam tikra dalis paskutinio klasterio paprastai prarandama, nes failo dydis nesidalija iš klasterio dydžio. Antra vertus, klasterių smulkinimas didina jų skaičių faile ir atitinkamai priėjimo operacijų skaičių, kai vyksta failo skaitymas ar rašymas į failą. Taip pat didėja FAT elementų skaičius ir atitinkamai reikiamo klasterio paieškos trukmė.

MS-DOS išorinė komanda **CHKDSK** leidžia sužinoti disko tomo parametrus, tame tarpe klasterio dydį ir jų skaičių.

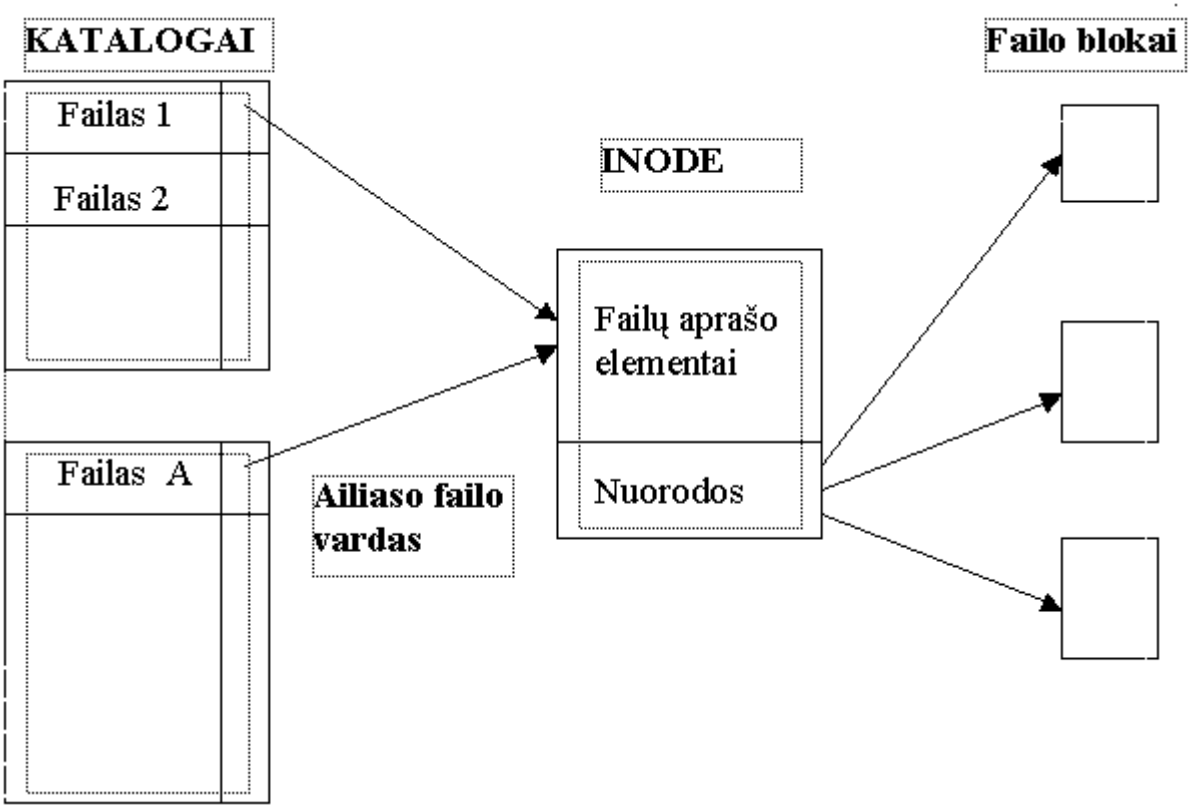
C:\DOSFILES>chkdsk c:

CHKDSK has NOT checked this drive for errors.
You must use SCANDISK to detect and fix errors on this drive.
Volume WIN98 created 2000.04.26 8:58
Volume Serial Number is 1404-3B6C

3 229 560 832 bytes total disk space
2 272 223 232 bytes available on disk
4 096 bytes in each allocation unit
788 467 total allocation units on disk
554 742 available allocation units on disk
655 360 total bytes memory
560 016 bytes free

8.5. UNIX failų sistema

Ryšiai tarp katalogų, mazgų (*Inode*) ir failų:

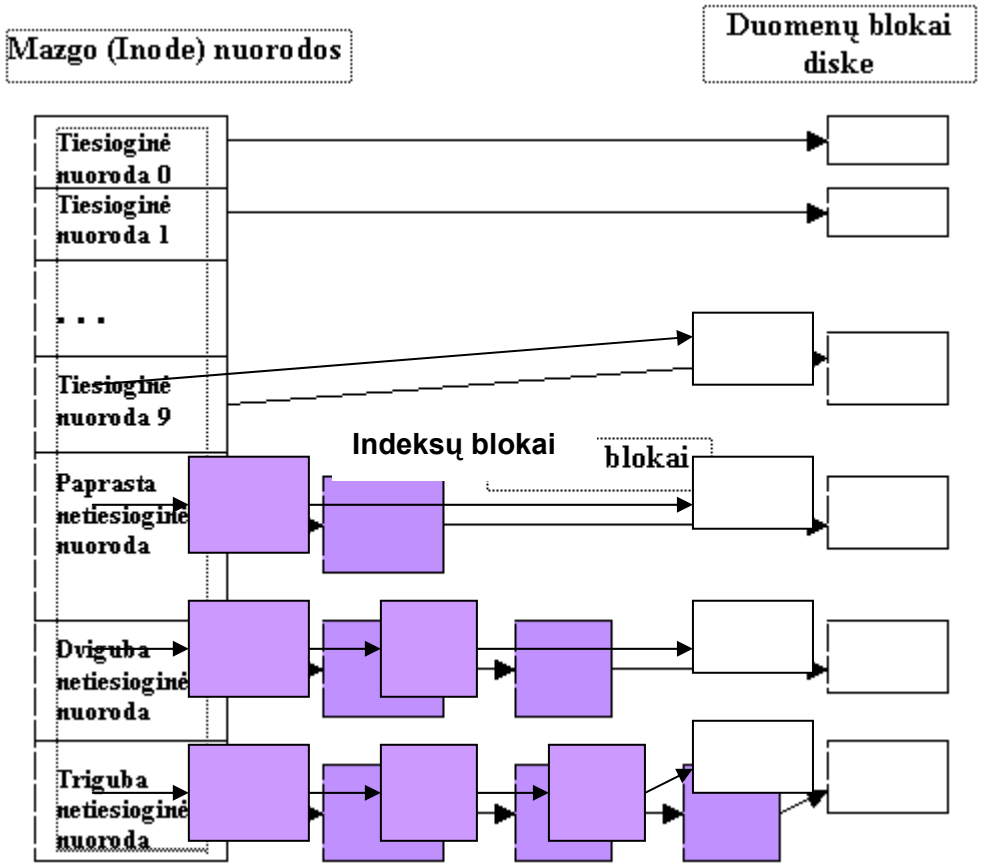


Informacija apie failus UNIX sistemoje saugojama atskirai nuo pačių katalogų specialiose struktūrose - *Inode* (mazguose). Kiekvieną failą atitinka vienas mazgas. Kataloguose saugojamas failo vardas ir nuoroda į atitinkamą mazgą. Visa kita informacija apie failą saugojama mazge (*inode*), tame tarpe, disko blokų, priklausančių failui, išdėstymas. Tipinė INODE struktūra:

Failo savininkas - vartotojo vardas (*owner of file*)	vitalij
Vartotojo Grupės vardas (*group ID*)	staff
Failo tipas (*file type*)	regular
Teisės (*permissions*)	rwxr-xr-x
Last accessed	10 May 2000 14:00
Last Modified	7 May 2000 11:15
Last Inode Modification	7 May 2000 11:15
Failo dydis (*File size*)	54400 bytes
Ryšių skaičius (*Number of links*)	…
Pointers to disk allocations	…

Informacija saugojama mazge gali būti peržiūrėta komanda ***ls –l.***

Mazgo nuorodų sistema. Mazge saugojama 13 nuorodų; 10 tiesioginių į blokus ir 3 netiesioginės, t.y. rodančios "indeksų blokus", kuriuose taip pat saugojamas nuorodų sąrašas.
Tad pirmosios 10 nuorodų adresuoja 512 * 10 = 5120 baitų,
11-oji – (jei blokas yra 512 baitų, o viena nuoroda jame užima 4 baitus, tai iš viso bloke yra 128 nuorodos) 5120 + 128*512 = 70656 baitų
ir t.t.



Tomo struktūra UNIX sistemoje.

Pakrovimo blokas (*boot block***)**. Pirmosios pakopos pakrovėjas, kuris perduoda valdymą **pagrindiniam sistemos pakrovėjui.**

Super Blokas (*super blocks***)–** nusako disko dydį, mazgų ir duomenų blokų išdėstymą diske.

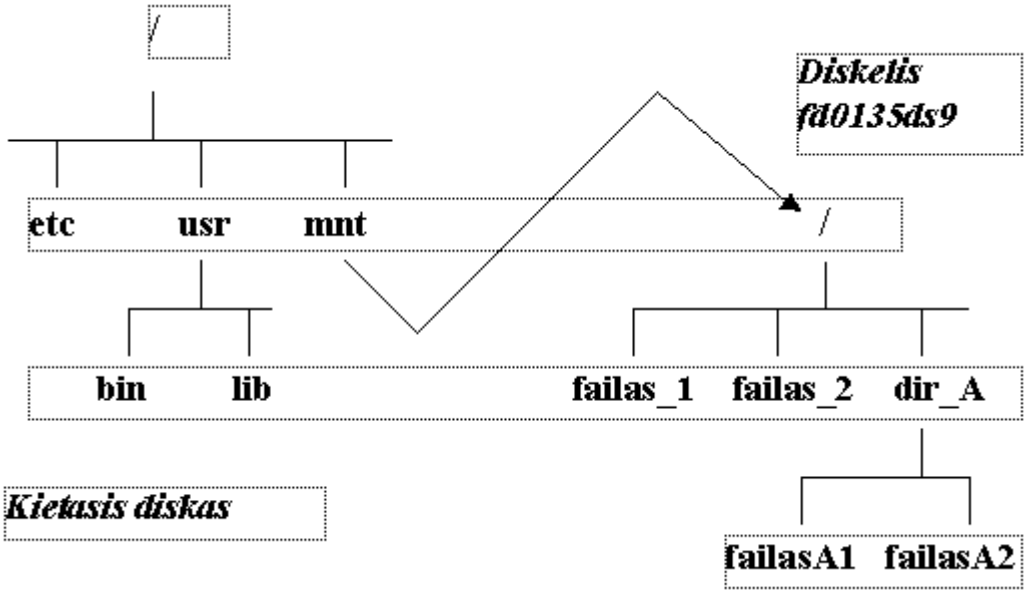
Mazgai (*Inodes***) –** disko erdvė mazgų saugojimui.

Duomenų blokai (*data blocks***) –** likusi disko erdvė, naudojama katalogų ir duomenų failų saugojimui.

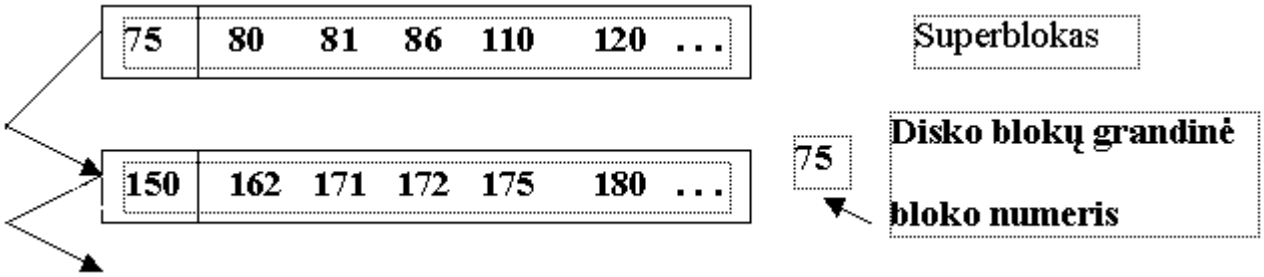
Kadangi diske yra fiksuotas mazgų skaičius, jis, atitinkamai, nusako maksimalų failų skaičių diske. Paprastai vartotojas gali pakeisti disko erdvės, skirtos mazgų saugojimui, dydį ir tokiu būdu padidinti ar sumažinti duomenų blokams skiriamą erdvę.

UNIX komanda ***df*** leidžia sužinoti tomo parametrus, tame tarpe mazgų ir duomenų blokų skaičių.

Tomo montavimas. UNIX Failų sistema gali susidėti iš daugiau nei vienos fizinės failų sistemos (tomo), t.y., kietų diskų, diskelių, skirtingų disko particijų. Jei MS-DOS (Windows) sistemoje kiekvienam tomui priskiriama atitinkamo "disko" raidę (A:, C:, F:, ir t.t.), UNIX sistemoje skirtingi tomai jungiami į egzistuojantį katalogų medį. Tam naudojamos komandos ***mount*** ir ***umount***, pvz., pateikta komanda jungia diskelio (vardu */dev/fd0135ds9*) failų sistemą prie disko katalogo */mnt*:



Laisvos vietos kontrolė UNIX sistemoje. *Programa* **mkfs** *sukuria surištų blokų grandinę; kiekvienas blokas saugoja laisvų duomenų blokų numerius ir nuorodą į tolesnį grandinės bloką; pirmas yra superblokas; išlaisvinti blokai pridedami prie grandinės galo.*



8.6. Pagrindinės Windows NT failų sistemos savybės

- Suderinamumas su MS-DOS ir OS/2
- Patobulintas failų pavadinimas ir Unicode failų vardai

- Suderinamumas su POSIX (portable UNIX) reikalavimais:
  - didžiųjų ir mažųjų raidžių atpažinimas failų varduose,
  - aliasų (hard link) naudojimas, t.y., failas gali turėti daug vardų,
  - informacijos apie paskutinius failų atidarymus saugojimas ir kt.

- Saugumas ir patikimumas:
  - atstatoma failų sistema (registruoja IO įvykius specialiame faile – *transaction log file*);
  - operatyvus klaidų ištaisymas (automatinis informacijos perkėlimas iš blogų disko paviršaus sektorių ir jų fiksacija);
  - loginių diskų skaidymas tarp fizinių (*disk striping*), t.y., vieną loginį atitinka keli fiziniai, kas didina aptarnavimo greitį;
  - *disk striping with parity* – naudojami 3 ar daugiau diskų, kurių vienas tarnauja kaip informacijos saugojimo etalonas. Tai leidžia atstatyti duomenis klaidos atveju. Tokia technika vadinasi *RAID* (*Redundant Array of Inexpensive Disks*- perteklinis nebrangių diskų masyvas).

- Dideli failų dydžiai. MS-DOS sistemoje failo dydžiui vaizduoti naudojami 32 bitai, t.y. max dydis yra 4 GB. NT naudoja 64 bitus failo dydžiui vaizduoti, t.y. . max dydis yra $2^{64} \approx 18*10^{18}$ baitų.

Diskinių sistemų spartos didinimas

- Blokavimas (tipinis bloko dydis – 512 baitų)

- *Disk cashing, cache* – tai buferių rinkinys, naudojamas dažniausiai prieinamos informacijos (blokų) saugojimui. Prielaida: dažniausiai naudojami blokai ir toliau bus dažniausiai naudojami. Vartotojas gali paspartini informacijos perkėlimą į diską iš *cache* atminties naudojant *sync()* sisteminę funkciją ar *sync* komandą.

- *RAM* diskai

- Failų reorganizacija ir defragmentacija.

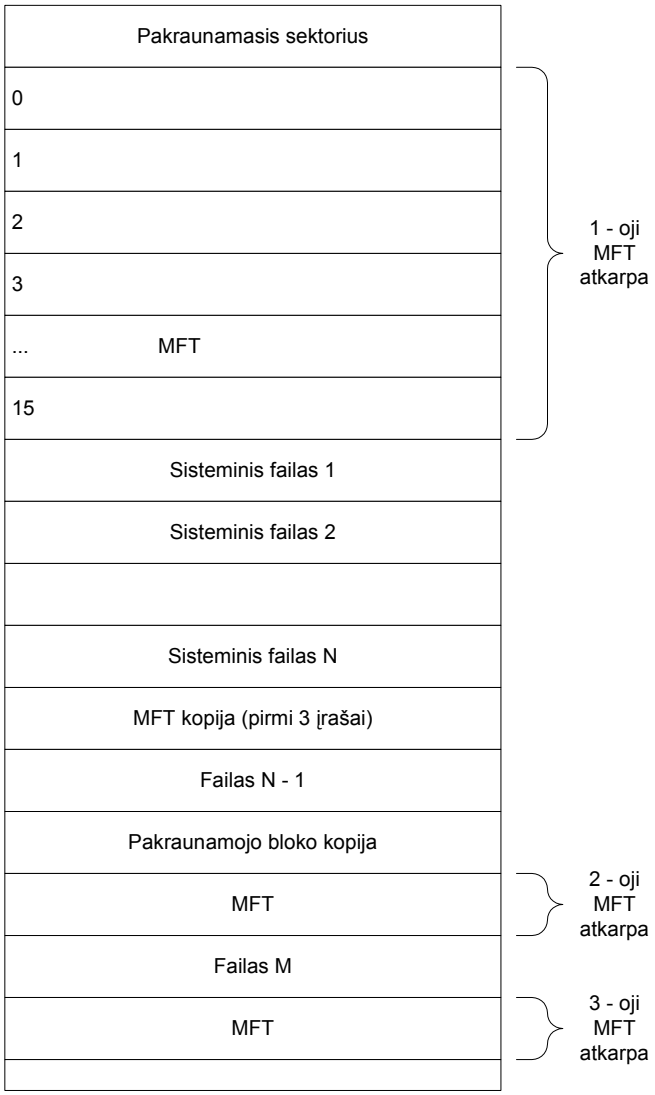
8.7. NTFS failų sistema

NTFS failų sistema buvo sukurta kaip pagrindinė operacinei sistemai Windows NT 90-tų metų pradžioje.

Pagrindinės savybės, kurios skiria NTFS nuo kitų sistemų yra:

1. palaiko didelius failus ir diskus iki 2^{64} baitų;
2. atstatoma „sugriuvus“ sistemai, programoms ir valdymo aparatūrai;
3. greitas operacijų kiekis, tame tarpe ir su dideliais diskais;
4. žemas fragmentavimo lygmuo;
5. lanksti struktūra, leidžianti vystytis pridedant naujus įrašus ir failų atributus, išsaugant suderinamumą su ankstesnėmis failų sistemomis;
6. pastovumas „sugriuvus“ disko kaupikliams;
7. ilgų simbolinių vardų palaikymas;
8. katalogų ir atskirų failų kontrolė;

8.7.1. NTFS tomo struktūra



Skirtingai negu FAT, NTFS tai arba failas, arba failo dalis. Pagrindinė NTFS tomo struktūra yra *pagrindine failų lentele (Master File Table, MFT)*. Kiekvienas MFT įrašas turi fiksuotą ilgį, priklausantį nuo disko dydžio, - 1, 2, arba 4 KB.

Daugumos šiandien naudojamų diskų MFT yra 2 KB.

Visi NTFS tomo failai yra apibrėžiami failo numeriu, kuris nusakomas failo pozicija MFT. Visas NTFS tomas susideda iš klasterių. Ankstesnėse failų sistemose į klasterius buvo skirstoma tik duomenų sritis.

Klasterio eilės numeris vadinamas *loginiu klasterio numeriu (Logical Cluster Number, LCN)*. NFTS failas taip pat susideda iš klasterių eilutės. Klasterio eilės numeris vidury failo vadinamas *virtualiuoju klasterio numeriu (Virtual Cluster Number, VCN)*.

NTFS failų sistemai bazinis disko vietos paskirstymo vienetas – ištisinė klasterių sritis vadinama *atkarpa*. NTFS atkarpos adresu naudojamas jo pirmojo klasterio loginis numeris, o taip pat klasterių kiekis k atkarpoje, t.y. pora (LCN, k). Tokiu būdu failo dalis patalpinta į atkarpą ir prasidedanti nuo virtualaus klasterio VCN, charakterizuojama adresu, susidedančiu iš trijų skaičių (VCN, LCN, k).

Pakraunamasis sektorius išdėstomas tomo pradžioje, o jo kopija tomo viduje. Pakraunamąjį sektorių sudaro standartinis BIOS parametrų blokas, blokų kiekis tome, o taip pat pagrindinės MFT kopijos pradinis loginis klasterio numeris ir veidrodinė MFT kopija.

Toliau išsidėsto pirma MFT atkarpa, turinti 16 standartinių įrašų apie sisteminius NTFS failus. Ji sukuriama formatuojant diską. Šių failų paskirtis:

Įrašo numeris	Sisteminis failas	Failo vardas	Failo paskirtis
0	Pagrindinė failų lentelė	$Mft	Turi pilną NTFS tomo failų sąrašą
1	Pagrindinės failų lentelės kopija	$MftMirr	Pirmųjų 3 MFT įrašų veidrodinė kopija.
2	Žurnalo failas	$LogFile	Tranzakcijų sąrašas, kuris naudojamas sistemai atstatyti „griovimo“ metu.
3	Tomas	$Volume	Tomo vardas, NTFS versija ir kita informacija apie tomą.
4	Atributų apibrėžimo lentelė	$AttrDef	Vardų, numerių ir atributų aprašų lentelė.
5	Šakninio katalogo indeksas	$.	Šakninis katalogas.
6	Klasterių bitų matrica	$Bitmap	Panaudotų tomo klasterių pažymėjimas
7	Pakraunamasis skirsnio sektorius	$Boot	Pakraunamojo skirsnio sektoriaus adresas
8	Blogų klasterių failas	$BadClus	Failas, kuriame surašyta informacija apie visus nustatytus tome blogus klasterius.
9	Kvotų lentelė	$Quota	Kiekvienam vartotojui skirtos panaudojamos vietos diske kvotos.
10	Registro simbolių pertvarkimo lentelė	$Upcase	Naudojama pertvarkant simbolių registrą Unicode kodavimui.
11-15	Rezervuoti naudojimui ateityje		

NTFS failas pilnai išdėstomas MFT lentelės įraše, jei tai leidžia padaryti jo dydis. Tuo atveju, kada failo dydis didesnis nei MFT įrašo dydis, įrašomi tik kai kurie failo atributai, o likusioji failo dalis patalpinama atskiroje tomo atkarpoje (arba keliuose atkarpose). Failo dalis išdėstoma MFT įraše, vadinama *rezidentine* dalimi, likusios dalis *nerezidentinės*. Adresinė informacija apie atkarpas, turinčias nerezidentines failo dalys, išdėstoma rezidentinės dalies atributose.

8.7.2. NTFS failų struktūra

NTFS tome kiekvienas failas ir katalogas susideda iš atributų rinkinio.

Kiekvienas NTFS failo atributas susideda iš laukų: atributo tipas, atributo ilgis, atributo reikšmė, ir gali būti, atributo vardas. Atributo tipas, ilgis ir vardas sudaro atributo antraštę.

Yra ir sisteminis atributų rinkinys. Sisteminiai atributai turi fiksuotus vardus ir jų tipų kodus, o taip pat apibrėžtą formatą. Gali būti taikomi ir vartotojo apibrežti atributai.

Sisteminis rinkinys susideda iš tokių atributų:

1. *atributų sąrašas (Attribute List)* – atributų sąrašas iš kurių susideda failas;
2. *failo vardas (File Name)* – šis atributas saugo ilgą failo vardą UNICODE formato, o taip pat įejimo numerį MFT lentelėje tevininiam katalogui, jeigu šis failas egzistoja keliuose katalogose, tai jis turės keletą atributų file name tipo. Šis atributas visada turi būti rezidentiniu.
3. *MS-DOS vardas (MS-DOS Name)* – šis atributas saugo 8.3 formato failo vardą;
4. *versija (Version)* – atributas saugo paskutinės failo versijos numerį;
5. *saugumo deskriptorius (Security Description)* – saugo informaciją apie failo apsaugą;

6. *tomo versija (Volume Version)* – tik sisteminiuose tomo failuose naudojama tomo versija;
7. *duomenys (Data)* – saugo įprastus failo duomenys;
8. *MFT bitų matrica (MFT bitmap)* – saugo blokų panaudojimo tome matricą;
9. *indekso šaltinis (Index Root)* – medžio „šaknis“ naudojama failų paieškai kataloguose;
10. *indekso išsidėstymas (Index Allocation)* - nerezidentines medžio indeksinio sąrašo dalys;
11. *standartinė informacija (Standart Information)* – saugo visa likusią informaciją apie failą.

NTFS failai, priklausomai nuo išsidėstymo būdo, skirstomi į nedidelius, didelius, labai didelius ir superdidelius.

8.8. Įvedimas-išvedimas (I/O)

8.8.1. I/O įtaisų įvairoves faktoriai

Charakteristika	Reikšmių diapazonas
Informacijos perdavimo greitis	Diskas: 1- 4 MB/s Klaviatūra: 10-20 B/s
Perdavimo vienetai	Diskas: 512/1024 B blokai Monitorius: simboliai, pikseliai
Operacijos	Diskas: skaitymas, rašymas, paieška Spausdintuvas: rašymas, popieriaus sukimas, šriftų nustatymas
Klaidų situacijos	Diskas: skaitymo-rašymo klaidos Spausdintuvas: nėra popieriaus

Kompiuteris (OS) bendrauja su I/O įtaisais per I/O šyną.. Kiekvieną įtaisą atitinka tam tikras kontroleris tiesiogiai sujungtas su šia šyna ir atsakingas už duomenų perdavimą į/iš operatyviosios atminties. Kiekvienam įtaisui taip pat priskirtas adresas, pagal kurį jį randa procesorius ir kt. I/O įtaisai dirba lygiagrečiai su procesoriumi (asinchroniškai) ir naudoja pertraukimus pranešimams apie I/O operacijos užbaigimą.

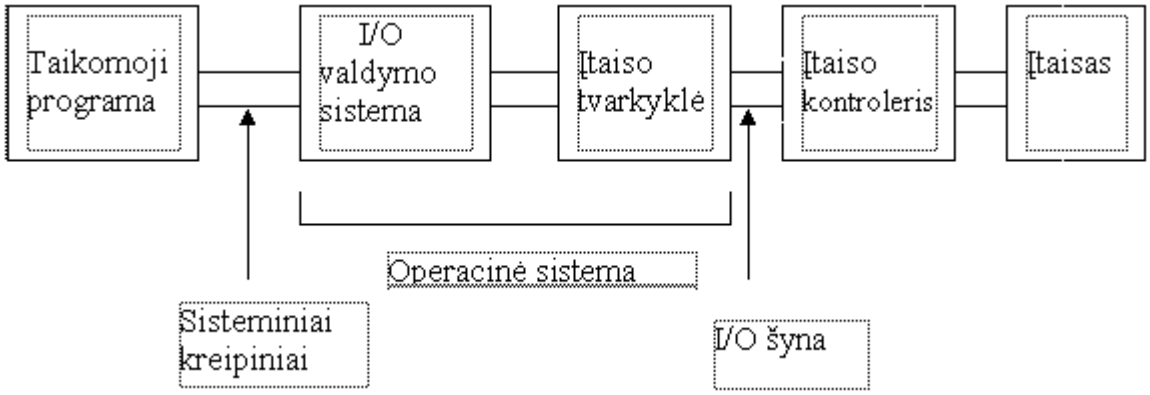
Kita technika – DMA (tiesioginis priėjimas prie atminties). Čia procesorius tik inicijuoja I/O operaciją, o toliau kontroleris bendrauja tiesiogiai su atmintimi. Nors šiuo atveju DMA “voginėja” (iš procesoriaus) atminties skaitymo ciklus, vis tiek laiko išlaidos yra žymiai mažesnes už pertraukimų apdorojimą.

8.8.2. I/O sistemos tikslai ir struktūra

Efektyvumas. Elektroninės operacijos (nanosekundės) *versus* mechaninės operacijos (sekundės ir milisekundės). Reikalavimas I/O sistemai: užtikrinti maksimalų I/O įtaisų efektyvumą, t.y., maksimalų duomenų *perdavimo* greitį vykdant naudingą darbą (plg. su procesų valdymu).

Bendrumas ir įtaisų nepriklausomumas. Vartotojai ir programuotojai paprastai atskirti nuo elementarių įtaisų operacijų (pvz., skaitymo galvutės pozicionavimas, reikiamo sektoriaus laukimas ir t.t.) tam tikrais programinės įrangos sluoksniais (lygiais). Netgi OS negali “žinoti” visų įmanomų įtaisų ir prietaisų darbo režimų, vidinių protokolų, signalų, mechaninių ir kitų operacijų ir pan.

I/O sistemos struktūra. Pagrindinių I/O sistemos tikslų įgyvendinimas nusako tokią jos struktūrą, kurioje vartotojas (viename gale) yra atskirtas nuo fizinių įtaisų (kitame gale) tam tikrais techniniai ir programiniais moduliais (sluoksniais) (žr. pav).



Taikomoji programa. Operuoja loginėmis instrukcijomis ir duomenų struktūromis (pvz., išvesti failą *x.txt* į spausdintuvą), kurios nepriklauso nuo fizinės įtaiso realizacijos.

I/O valdymo sistema (IOCS). Atlieka pirminį sisteminio kreipinio apdorojimą ir nukreipia jį tolimesniam tvarkymui. Apdoroja visus I/O pertraukimus.

Įtaisų tvarkyklės (drivers). Pagrindinis uždavinys – loginių kreipinių vertimas į specifines komandas, skirtas pačiam įtaisui (arba tam tikram įtaisų tipui). Pvz., jei taikomoji programa pareikalavo rašymo į diskelį, tvarkyklė atliks tokius konkrečius veiksmus: patikrins ar diskelis yra įdėtas, lokalizuos failą kataloge, nukreips galvutes į reikiamą takelį ir pan.

Tvarkyklė – tai tam tikra OS dalis, bet priklausomybės nuo OS pobūdis gali būti gana skirtingas. Taip UNIX sistemoje tvarkyklių kodas paprastai yra kompiliuojamas ir sujungiamas su sistemos branduoliu. MS-DOS ir (iš dalies) Windows sistemose tvarkyklės instaliuojamos ir pakraunamos dinamiškai. Pirmuoju atveju sistema yra kiek efektyvesnė ir pasižymi paprastesne struktūra, bet naujo įtaiso pajungimas paprastai reikalauja branduolio regeneracijos. Antrasis būdas žymiai paprastesnis vartotojui, be to, tik šiuo momentu reikalingos tvarkyklės kraunamos į atmintį.

Įtaisų kontroleriai. Derina konkrečios kompiuterinės sistemos, jos I/O šynos ir konkretaus įtaiso charakteristikas.

Įtaisai (ypač išoriniai) paprastai gali būti naudojami su skirtingomis OS.

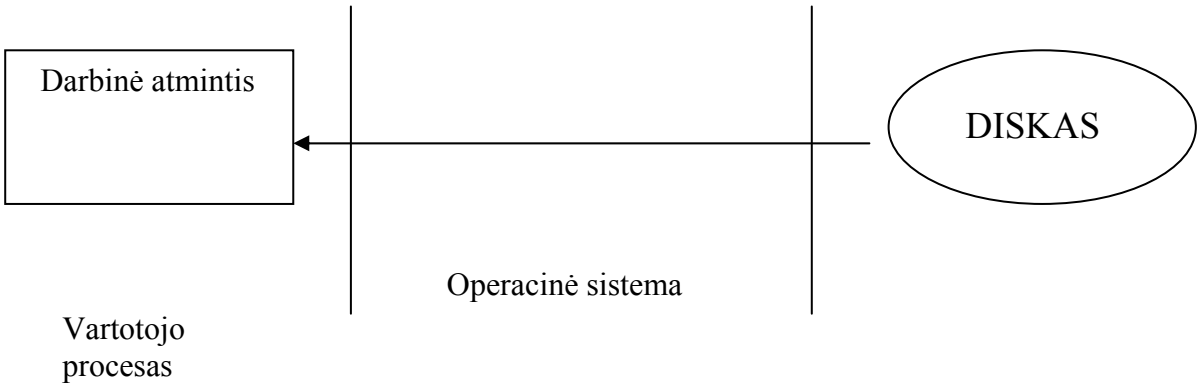
Blokiniai ir simbolių įtaisai. Blokiniai įtaisai perduoda duomenis grupėmis (blokais), tuo tarpu simbolių įtaisai perduoda po vieną simbolį per laiko tarpą. Paprastai išorinės atminties (magnetiniai) įtaisai yra blokiniai, o visi kiti – simbolių. Blokiniai įtaisai yra sudėtingesni ir reikalauja daugiau OS servisų, pvz., tokių kaip klaidų patikrinimas, abipusis duomenų perdavimas, atsitiktinis priėjimas prie duomenų.

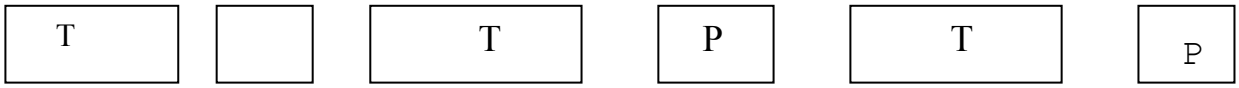
Virtualūs įtaisai. Tai realaus įtaiso imitavimas operacinėje sistemoje. Vartotojas įsivaizduoja jį kaip realų įtaisą, kuris normaliai reaguoja į visus sisteminius kreipinius. Pvz., spausdintuvo "spuleris" (spooler) saugoja spausdintus duomenis specialiame faile, o realus spausdinimas įvyksta tada, kai tai patogu operacinei sistemai.

Privalumai: - spausdinimas gali vykti lygiagrečiai su kitais procesais; - galima naudoti daugiau išorinių įtaisų, nei fiziškai yra; - pasiekiamas įtaisų nepriklausomumas.

Buferiai. Tai pagrindinė technika, skirta I/O operacijų efektyvumui padidinti.

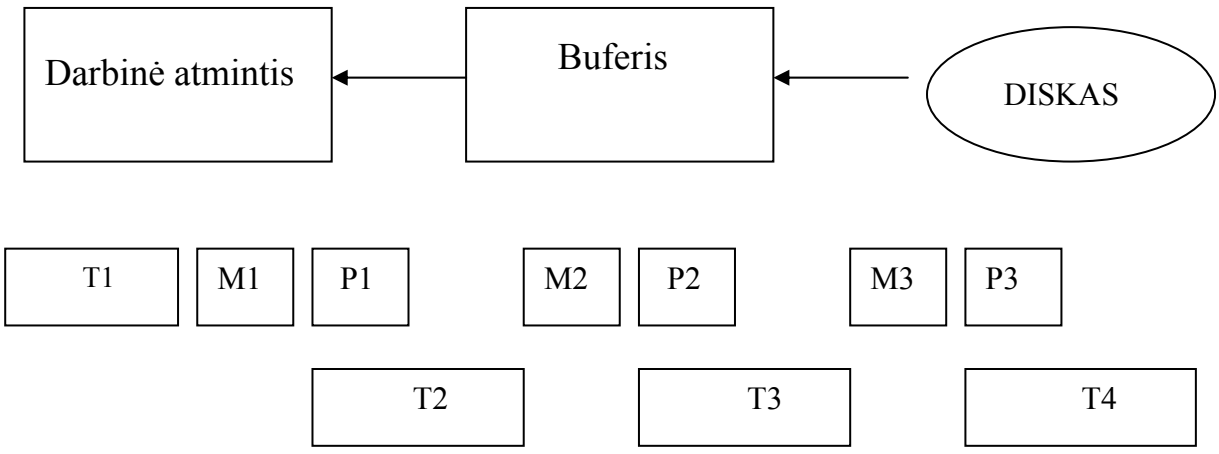
Iliustracija: tam tikras procesas nuosekliai skaito duomenų blokus iš disko. Procesas inicijuoja sisteminius kreipinius tam, kad nuskaitytų pirmą duomenų bloką ir perduotų jį į darbinę proceso atmintį. Kai blokas atsiranda atmintyje, procesas atlieka duomenų apdorojimą, kurio metu I/O operacijos neatliekamos. Kai apdorojimas baigtas, inicijuojamas antrojo bloko skaitymas ir t.t.:





Duomenų perdavimas (T – *transfer*) ir apdorojimas (P - *processing*) be buferio.

Bendras laikas = T + P

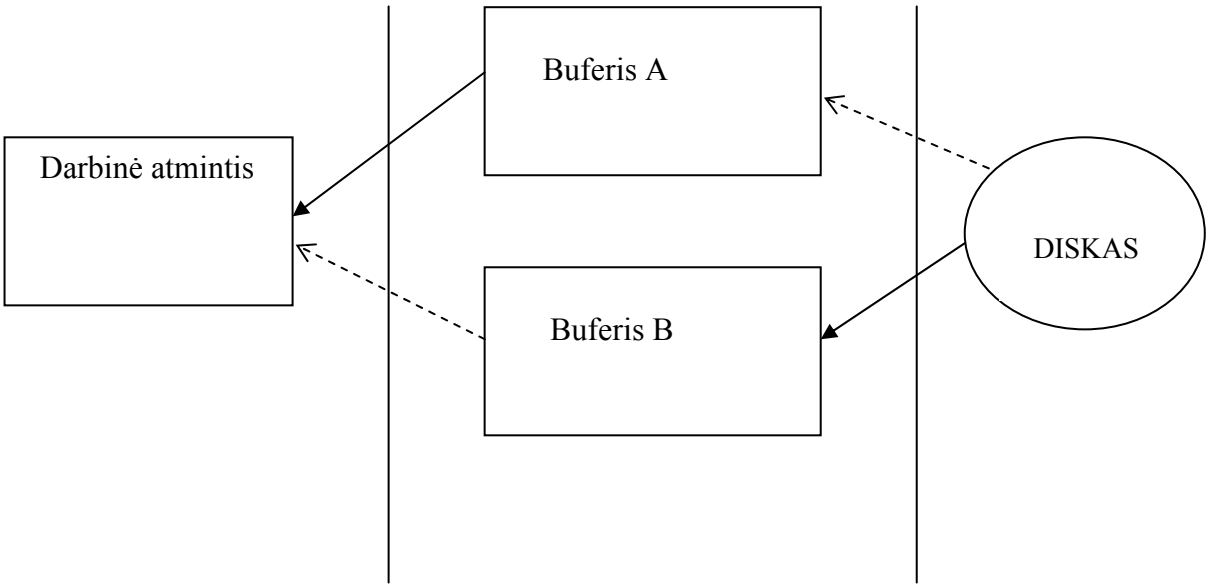


Perdavimas (T) į buferį, perkėlimas į atmintį (M - *moving*) ir apdorojimas (P).

Bendras laikas = T + M, ir M << P

Dvigubi buferiai.

Iš vieno buferio informacija gali būti keliama į atmintį, tuo tarpu kai vyksta rašymas į kitą buferį:



8.8.3. UNIX įvedimo-išvedimo sistema

Įtaisai ir jų vardai. Vienas iš UNIX kūrimo tikslų buvo sudaryti unifikuotą I/O operacijų sistemą. Taigi, UNIX branduolys visus įtaisus ir failus traktuoja vienodai. Įtaisai vaizduojami specialiais failais, kurie atsiranda tam tikruose sistemos kataloguose.

Kita svarbi UNIX I/O sistemos savybė – duomenų perdavimas tarp procesų ir failų ar įtaisų vykdomas kaip baitų srautas (t.y., neturi struktūros ir tipo).

Kataloge ***/dev*** saugojamas specialių failų, atitinkančių sistemos įtaisus, sąrašas. Jį galima peržiūrėti tokia komanda:

$ ls –l /dev

Tipinis rezultatas:

crw------- 1	vitalij	terminal	0	0	Apr 12	10:05	tty01
brw-rw-rw- 4	bin	bin	2	60	Apr 15	14:20	fd0

kur pirmas atributas vaizduoja simbolinį (c) arba blokinį (b) įtaisą.

Kai kuriose sistemos versijose specialūs failai grupuojami pagal įtaisų tipus ir vardus. Tipiniai vardai yra tokie:

/dev/console	sistemos konsolė
/dev/tty01	vartotojo terminalas 1
/dev/tty02	vartotojo terminalas 2
/dev/rmt0	magnetinė juosta
/dev/dsk/0s0	kietas diskas 0
/dev/dsk/f03h	1.44 MB diskelio įtaisas
/dev/lp	spausdintuvas (line printer)

Skirtingai nuo įprastų duomenų failų, kurie siejami su diske saugojama informacija, įtaisų vardai asocijuojami su atitinkamomis tvarkyklėmis.

Kai UNIX sistemoje reikia pridėti naują įtaisą, atitinkamos tvarkyklės kodas yra kompiliuojamas ir sujungiamas su sistemos branduoliu. Tai įprasta šioje sistemoje procedūra. Kai kurios tvarkyklės naudojamos virtualiųjų įtaisų sukūrimui. Pvz., ***/dev/pty*** reiškia “pseudo terminalą”, kuris naudojamas nutolusio vartotojo terminalo emuliavimui.

Terminalai. Istoriškai vartotojo terminalai yra labai svarbi I/O sistemos dalis. Jie identifikuojami ***tty*** vardais ***/dev*** kataloge. Susidoroti su didele terminalu įvairove naudojamos specialios technikos. Tam tikrose sistemose naudojamas failas ***/etc/termcap*** , kuriame užkoduotos kiekvieno sistemos terminalo charakteristikos. Šis failas naudojamas programų, kurios nori pasinaudoti išplėstinėmis terminalinio įtaiso funkcijomis (pvz., teksto redaktorius talpina žymeklį į tam tikrą ekrano poziciją). Kitose sistemose informacija apie terminalus saugojama atskiruose failuose: ***/usr/share/terminfo.***

Darbo metu pageidautina informuoti sistemą apie savo terminalo tipą. Tam sisteminiam kintamajam TERM reikia priskirti terminalo kodą, pagal kurį bus surastas reikiamo terminalo aprašymas. Pvz.;

$ TERM = ansi
$ export TERM

8.8.4. MS-DOS ir Windows I/O sistemos

ROM BIOS (arba "firmware") – tai MS-DOS operacinės sistemos dalis, atsakinga už žemio (fizinio) lygio I/O operacijų vykdymą. Aukštesnio lygio paslaugas suteikia sistemos branduolys. Pvz., BIOS gali rašyti į tam tikrus fizinius sektorius diske ir nieko nežino apie tokias logines struktūras kaip failai ir katalogai, kurie aptarnaujami iš DOS'o. Tiek BIOS, tiek DOS servisai prieinami programuotojui per sisteminius kreipinius. MS-DOS sistemoje IOCS tiksliai atitinka objektinis sisteminis failas IO.SYS, o sistemos branduolį MSDOS.SYS.

Standartiniai įtaisai ir tvarkyklės. Tipiniai standartinių įtaisų MS-DOS sistemoje vardai:

con:	klaviatūta/monitorius
com1:	nuoseklusis uostas 1 (komunikacijos)
com2:	nuoseklusis uostas 2
lpt1:	spausdintuvo uostas 1 (lygiagretasis uostas)
lpt2:	papildomas spausdintuvo uostas
prn:	loginis spausdintuvo uostas (paprastai atvaizduojamas į ***lpt1:***)
A:	pirmas diskasukis
B:	antras diskasukis
C:	pagrindinis kietasis diskas
D: E: ir t.t.	papildomi kieti ar kompaktiniai diskai, arba "particijos", t.y., loginiai diskai.

MS-DOS ir Windows sistemose tvarkyklės talpinamos į atmintį dinamiškai, t.y., kai sistemos startuoja arba yra perkraunamos. MS-DOS sistemoje tvarkyklės – tai binariniai failai (dažniausiai .sys tipo), kurie aprašomi sisteminiame konfigūracijos faile CONFIG.SYS naudojant komandą ***DEVICE = tvarkykle.sys***, pvz.,

. . .
DEVICE = C:\HIMEM.SYS
DEVICE = C:\DOS\DISPLAY.SYS CON =(EGA,437,1)
. . .

Windows sistemoje tvarkyklės įgyvendinamos kaip dinamiškai susietos bibliotekos. Tai leidžia turėti atmintyje tik vieną tvarkyklės kodo kopiją daugeliui procesų. Be to įtaisų gamintojai gali realizuoti įvairias tvarkykles (net vienam įtaisui) nekeičiant pačios sistemos kodo. "Plug and Play" standartas leidžia atlikti dinaminį sistemos rekonfigūravimą ir automatiškai pridėti naują įtaisą bei įdiegti jo tvarkyklę.

Literatūra

1. V. Denisovas Sisteminis programavimas, Klaipėdos universitetas, 2000.
2. К.Рейчард, Э.Фостер – Джонсон, UNIX Справочник, Питер, 1999.
3. Н. А. Олифер, В. Г. Олифер, Сетевые операционные системы, Питер, 2001.
4. А. Соловьев, Программирование на Shell (Unix), Nevod Ltd.
5. Maurice J. Bach, THE DESIGN OF THE UNIX OPERATING SYSTEM, 1998