1.1 Mirties taškas. Jo susidarymo sąlygos. Resursų priskyrimo grafai ir mirties taško situacija

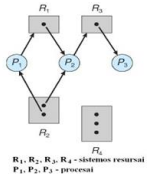
Mirties taškas sistemoje - tai tokia situacija, kai ilgam laikui yra užblokuojama grupė procesų, kurie varžosi dėl tų pačių sistemos išteklių arba komunikuoja su kitais, užblokuotais procesais.

Mirties taškas kyla kai du ar daugiau procesų turi konfliktuojančius poreikius.

Mirties taškas sistemoje tampa galimu, jei sistemoje susidaro šios sąlygos:

- **Tarpusavio išskirtinumo**
- **Turėjimo ir laukimo**
- **Neperėmimo**
- **Ciklinio laukimo sąlyga**

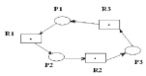
Resursas gali turėti keletą identiškų elementų (du spausdintuvai), grafe tai atžymima keliais taškais **R** tipo dėžutėje (R2, **R4** resursas).



Jei procesui resursas yra priskirtas, tai rodyklė nukreipta nuo resurso į procesą (R1 yra priskirtas procesui **P2).**

Jei procesas reikalauja resurso, tai rodyklė nukreipta nuo proceso į resursą (P1 reikalauja **R1).**

Sistema yra mirties taško situacijoje, nes kiekvienas procesas yra užsiėmęs resursą, kurio prašo kitas procesas ir joks procesas nenori atiduoti resurso, kurį jis valdo.



1.2. Mirties taško prevencija:
a) tiesioginiai būdai (jų galimybės, taikomi metodai)
b) netiesioginiai būdai

Netiesioginiams mirties taško sutrukdymo metodams būdinga tai, kad jie neleidžia susidaryti vienai iš trijų paminėtų sąlygų.
Tiesioginiai metodai sutrukdantys mirties taško susidarymui paprastai neleidžia susidaryti cikliniam laukimui.

Netiesioginiai metodai yra orientuoti į pirmas tris sąlygas:

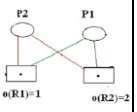
Tarpusavio išskirtinumo sąlyga Šios sąlygos egzistavimo dažnai negalima suardyti,

Turėjimo ir laukimo sąlyga
Šią sąlygą galima suardyti, reikalaujant kad procesas visų jam
reikalingų resursų užprašytų iš karto .Tokio metodo minusai:

- Gali gautis **badavimo** situacija.
- **žemas resursų panaudojimas.**
- procesas turi žinoti, kokių resursų ir kiek jų jam reikės.

Neatėmimo sąlyga.
Ši sąlyga gali būti sutrukdoma keliais būdais.
Metodo esmė yra ta, kad iš proceso yra atimami jam anksčiau išskirti resursai.

Tiesioginiai:
Pateiksime protokolą, kurio laikantis galima ciklinio laukimo prevencija:
Nustatoma griežta tiesinio surikiavimo eilė **O()** įvairiems resursų tipams. Pavyzdžiui:



Procesui gali reikėti įvairių resursų tipų. Viena užklausa jis
gali užsiprašyti kelių vieno kažkurio resurso, tarkim **Ri,** vienetų.

Procesas, kuriam išskirtas resursas **Ri** gali užprašyti ir gauti
kelis resurso **Rj** vienetus tik tuo atveju, jei **O(Rj)>O(Ri).**
Šio reikalavimo prisilaikant **nesusidarys** ciklinis laukimas.

1.3. Mirties taško vengimas. Bankininko algoritmas, jo taikymo pavyzdys. Saugaus būvio nustatymo algoritmas, jo taikymo pavyzdys

Šiuo atveju leidžiama įvykti pirmoms trims sąlygoms, tačiau vykdomas protingas resursų išskyrimas, kuriam esant mirties taškas niekad nebus pasiektas.
Taikant šį metodą yra leidžiama didesnė konkurencija dėl resursų nei prevencijos atveju. Taikomi du būdai:

- Procesas nepradedamas, jei jo resursų užklausa gali vesti į mirties taško susidarymą.
- Resursų padidinimo užklausa nevykdoma, jei toks išskyrimas ves prie mirties taško susidarymo. Abiem šiais atvejais iš anksto turi būti nusakomi maksimalūs proceso poreikiai resursams.

Procesai yra kažkuo panašūs į vartotojus, norinčius pasiskolinti pinigų (resursų) banke. Bankininkas turėtų neduoti paskolos, jei jis negali patenkinti visų vartotojų poreikių.

Bet kuriuo momentu sistemos būvis gali būti apibrėžiamas **R(i), C(j,i)** reikšmėmis, kur **i** žymi **R** resurso tipą, o **j** procesą.
Matricos **A** elementas **A(j,i)** parodo, koks **i** –tojo resurso kiekis jau yra skirtas **j-tajam** procesui. (visoms **j,i** reikšmėms).
Sumarinis esamas laisvas **i** –tojo resurso kiekis yra nusakomas vektoriaus **V(i)-taja** komponente:
Matricos **N** elementai **N(j,i)** nusako, kiek **i-to** resurso reikia tam, kad **j-tasis** procesas galėtų baigti įvykdyti savo užduotį.

Sprendžiant, ar gali būti patenkinama proceso užklausa resursui, bankininko algoritmas testuoja, ar užklausos patenkinimo rezultate sistemos būvis gausis **saugus.** jei skyrus resursą procesui rezultate gausis saugus būvis, tai užklausa yra tenkinama, priešingu atveju ji nėra tenkinama.

Būvis yra laikomas saugiu, jei galima sudaryti tokią procesų seką **{P1...Pn},** kuriai esant bus galima kiekvienam iš sekoje esančių procesų priskirti visus to proceso įvykdymui reikalingus resursus. Esant saugiam būviui visada visi procesai bus pilnai įvykdyti.

Inicializacija:
Visi procesai skelbiami nebaigtais.
Darbiniam vektoriui W(i) priskiriame resursų kiekius
W(i)=V(i) visiems i;
Kartojama:
leškomas toks neužbaigtas procesas j, kuriam:
N(j,i)<=W(i), visiems i.
Jei tokio j near - pereiti I EXIT.
Jei toks j yra "užbaigti" šį procesą ir grąžinti to proceso užimtus resursus - skelbiant juos laisvais.
W(i)=W(i)+A(j,i) visiems i
Gįžti į kartojimą

EXIT: jei visi procesai tapo "užbaigtais", tai būvis laikomas saugiu, priešingu atveju - nesaugiu.

1.4. Mirties taško nustatymo algoritmas, jo taikymo pavyzdys Išėjimas iš mirties taško

Šiuo atveju procesų užklausos resursams visad yra tenkinamos (jei tik užtenka resursų - tai yra jei tik galima). Operacinei sistemai tokiu atveju reikia:

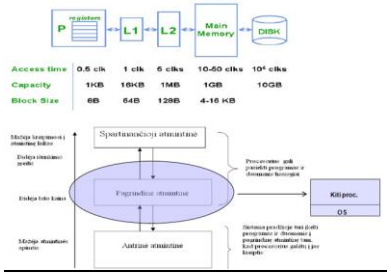
- *algoritmo patikrinimui*, ar nėra susidaręs mirties taškas.
- algoritmo, kuris nusakytų, *kaip išeiti iš šio mirties taško.*

Tikrinimas, ar yra susidaręs mirties taškas gali būti atliekamas kartu su kiekviena užklausa resursams.
Aišku, toks dažnas tikrinimas vartos daug CPU laiko.

Naudosime tas pačias matricas bei vektorius kaip ir resursų priskyrimo algoritme.
Bus atžymimi visi procesai, kurie nėra įėję į mirties taško situaciją.
Pradžioje visi procesai yra nepažymėti.
Atliekama:
Atžymimas kiekvienas j-tas procesas, kuriam išskirtas resursų kiekis **A(j,i)=0,** kiekvienam **i-tajam** resurso tipui (kadangi šie procesai near mirties taške).
Nustatomas darbinis vektorius **W(i)=V(i),** visoms **i** reikšmėms.
Kartojama:
Randamas nepažymėtas procesas **j,** kuriam **Q(j,i)<=W(i)** visoms **i** reikšmėms. Sustojama, jei tokio **j** proceso nėra.
Jei toks **j** procesas yra: **j-tasis** procesas pažymimas ir nustatoma **W(i)=W(i)+A(j,i),** visoms **i.**
Grįžtama į kartojimą.
Gale: kiekvienas nepažymėtas procesas yra mirties taške.

Aptikus susidariusį mirties tašką žudomas procesas, esantis mirties taško situacijoje.
Panaudojamas procesų suspendavimo/atnaujinimo mechanizmas
Pavyzdžiui, procesai iškeliami į swap sritį
Procesų vykdymas grąžinamas atgal:
Reikia išsaugoti tam tikrų kontrolinių taškų kontekstą
Neprarandamas su procesu atliktas darbas

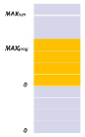
2.5 Atminties hierarchija ir charakteristikos



2.6 Reikalavimai pagrindinės atmintinės valdymui. Pagrindinės AV sąvokos

Patalpinimo vietos pakeitimo galimybė
Apsauga
Dalinimasis
Loginė organizacija
Fizinė organizacija

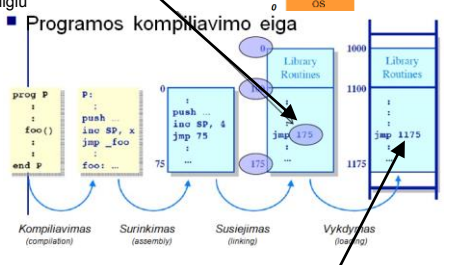
Fizinė adresų sritis – techninės įrangos palaikoma adresų sritis
Nuo 0 iki *MAXsys*



Loginė/virtuali adresų sritis – procesui matoma atminties sritis
Nuo *0* iki *MAXprog*

Fizinis (absoliutusis) adresas, loginis (bazinis, reliatyvus) adresas

Reliatyvus adresas – išreiškiamas bazinio adreso atžvilgiu



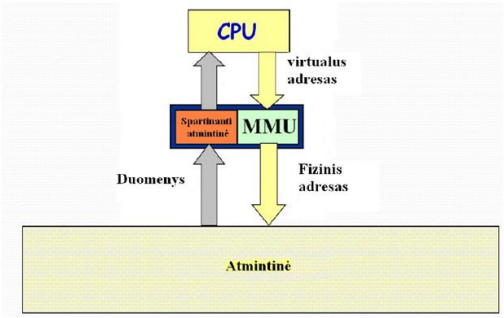
Fiziniai adresai – priskiriami programos vykdymo metu

2.7 Programos kompiliavimo etapai, jų paskirtis, sprendžiami uždaviniai

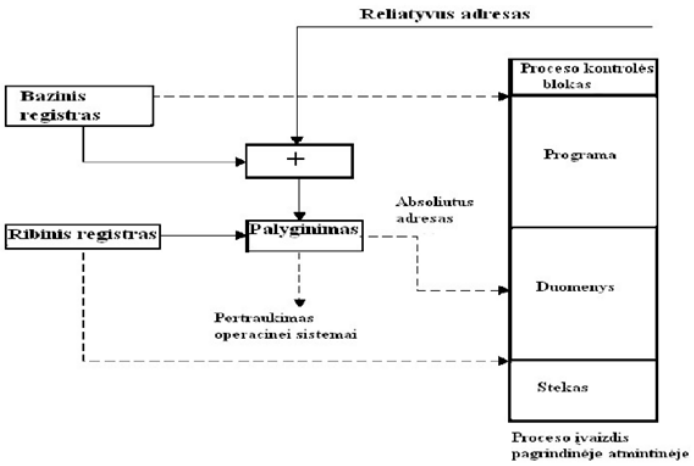
Atsakyta paveikslėlyje (2.6 klausime)

2.8 Adresų transliacijos schemos

Loginio adreso transliavimas į fizinį



Reliatyvaus adreso transliacija į fizinį adresą



2.9 Rezidentinės proceso dalies sąvoka

Į pagrindinę atmintinę įkelta proceso dalis dar yra vadinama **rezidentine** arba **darbine** dalimi.
Kodėl gi proceso vykdymui pakanka mažesnės srities nei viso proceso dydis?

Tai surišta su *lokališkumo* principu, kuris pasireiškia tuo:
Dažniausiai didelę proceso vykdymo laiko dalį procesas vykdo nedidelę komandų aibę (vyksta ciklas)
Naudoja greta esančius duomenis,
Dauguma skaičiavimų yra vykdomi nuosekliai.

Todėl bet kuriuo momentu procesui pakanka nedidelės rezidentinės srities pagrindinėje atmintinėje.

3.10 Atmintinės valdymo sprendžiami uždaviniai

Sprendžia problemas, kurių atsiranda keliant procesus iš antrinės atmintinės į pagrindinę:

Kurią proceso dalį įkelti ir kada?
Kur patalpinti įkeliamą procesą ar jo dalį?
Kuriuos duomenis iškelti, kad būtų daugiau vietos kitiems, įkeliamiems procesams?

3.11 Proceso patalpinimo pagr. atmintinėje būdai

Nuoseklių (ištisinių adresų) zonos skyrimas

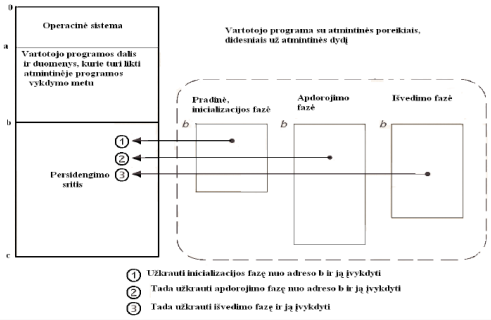
- procesas egzistuotų kaip vientisas, nuoseklių adresų erdvėje esantis blokas.
- tai labai paprastas atmintinės dalinimo būdas,
- atsiranda problemos:
  - tinkamo dydžio laisvo bloko atradimas
  - netinkamas atmintinės panaudojimas.

Neištisinės srities skyrimas

- procesas, jo duomenys skaldomi tam tikro dydžio, atskirose atmintinės vietose talpinamais gabalais (puslapiais, segmentais).
- lengviau atrasti tinkamas proceso patalpinimui vietas atmintinėje,
- leidžia padidinti procesų, vienu metu esančių pagrindinėje atmintinėje kiekį,
- realizacija yra sudėtingesnė.

3.12 Perdengimo mechanizmas

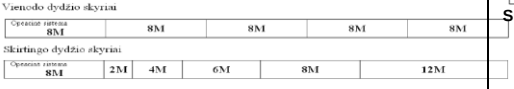
Ištisinė sritis ir *perdengimo* mechanizmas



3.13 Fiksuoto dydžio skyriai, jų pliusai, minusai

Prireikė algoritmų, kurie leistų paskirstyti atmintinę kelioms programoms (keliems procesams). Skirstant pagrindinę atmintinę yra naudojamas vienas iš algoritmų (Fiksuoto dydžio): Pagrindinė atmintis yra sudaloma į eilę nepersidengiančių skyrių (dalių). Šios dalys gali būti tiek vienodo tiek skirtingo dydžio.

- Procesas, kurio dydis yra mažesnis arba lygus skyriaus dydžiui, gali būti patalpinamas į šį skyrių.
- Procesorius gali greitai persijungti tarp procesų.
- Naudojami keli ribiniai registrai apsaugai nuo to, kad procesai negadintų vienas kito duomenų ar programos, kreipdamiesi į ne jam skirtą atmintinės bloką – tokie kreipiniai neleidžiami.
- Jei visi skyriai yra užimti, operacinė sistema gali iškelti (swap) procesą iš jo užimamo skyriaus.

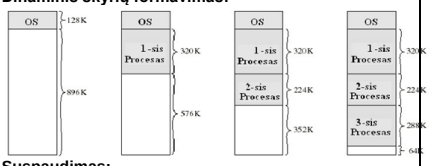


Atsiranda vidinės fragmentacijos problema:
nes nežiūrint kokia maža programa būtų – jai skiriamas visas skyrius ir jame gali būti daug nenaudojamos vietos.
Skirtingo fiksuoto ilgio skyriai kiek sumažina šią problemą, tačiau

3.14 Dinaminis skyrių formavimas: pliusai, minusai. Talpinimo algoritmai dinaminių skyrių atveju (taikymo pavyzdžiai), algoritmų pliusai, minusai

Taikant šį principą skyrių *kiekis,* jų *dydis* yra kintami. Kiekvienam procesui jį talpinant pagrindinėje atmintinėje yra išskiriamas *tokio dydžio skyrius, kokio jis prašo.*

Dinaminis skyrių formavimas:



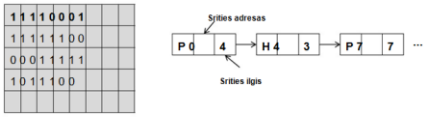
Suspaudimas:

4.15 Informacijos apie užimtas ir laisvas sritis saugojimas

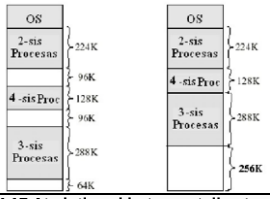
Dvejetainiai žemėlapiai;

- Didelę reikšmę turi srities dydis;
- Patogu naudotis

Surišti sąrašai;



problema išlieka.



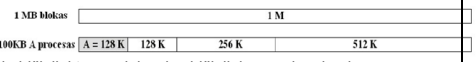
4.16 Atminties dalinimas naudojant „bičiuliškos sistemos“ algoritmą (taikymo pavyzdys)

Bičiuliška sistema, tai algoritmas, kuriuo bandoma apeiti tiek fiksuotų, tiek dinaminių skyrių problemas.

- Modifikuota šio algoritmo versija yra naudojama UNIX SVR4 .
- Atmintinės blokai, kurie yra išskiriami procesams yra **2^{K}** dydžio

Algoritmo žingsniai

- Pradžioje visa atmintinė yra laisva, taigi pradedama turint **2^{U}** dydžio bloką. Tarkime, atsiranda pareikalavimas patalpinti **S** dydžio procesą.
  - Jei **2^{U-1} < S <= 2^{U}**, tai yra išskiriamas visas blokas **2^{U}.**
  - Priešingu atveju blokas sudalomas į dvi vienodo dydžio **2^{U-1}** dalis (bičiulius).
    - Jei **2^{U-2} < S <= 2^{U-1},** tai procesui išskiriama viena iš dalių (vienas bičiulis), o jei ne, tai viena iš dalių vėl yra daloma į dvi dalis.
    - Šis procesas yra kartojamas tol, kol gaunamas mažiausias blokas, kuris yra lygus arba didesnis nei **S**.
    - Pavyzdys: Turim 1MB= 2^{10}KB=1024KB . Procesas prašo 100 KB
    - 2^6=512<100<2^{10}=1024 – netinka
    - 2^8=256<100<2^9=512 –netinka
    - 2^7=128<100<2^8=256 –netinka
    - 2^6= 64<100<2^7=128- tinka, skiriam 128 KB



Jei du bičiuliai tampa laisvais , bičiuliai yra apjungiami.
Operacinė sistema palaiko keletą sąrašų apie esančias skyles.

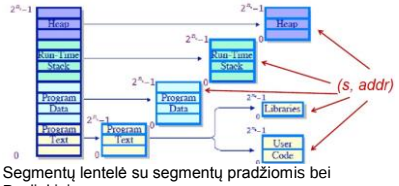
- **i-tas** sąrašas apima skyles, kurių dydis yra **2^{i}.**
- Kai tik pora bičiulių atsiranda **i-tame** sąraše, jie yra išmetami iš šio sąrašo ir apjungiami į vieną bendrą skylę **(i+1)** sąraše.

Atsiradus naujai užklausai, t.y. norint patalpinti **k** dydžio procesą, tokį:

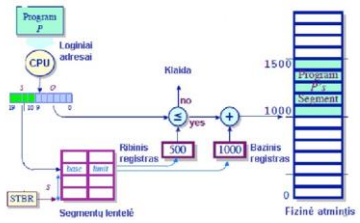
- kuriam galioja: **2^{i-1} < k <= 2^{i}**
- yra patikrinamas **i-tas** sąrašas. Jei šis sąrašas yra tuščias, tikrinamas **(i+1)** sąrašas.
- Jame radus skylę ji bus sudaloma į du bičiulius, viena iš dalių bus priskirta procesui, o kita įtraukta į **i-tą** sąrašą.

4.17 Atminties skirstymas taikant segmentacijos principus (pliusai, minusai), adreso transliavimo

Procesas išskaidomas loginiais segmentais
Kiekvienas segmentas apibrėžiamas adresu, nurodančiu segmento pradžią bei ribiniu poslinkiu.



Segmentų lentelė su segmentų pradžiomis bei Poslinkiais:



Išorinė fragmentacija egzistuoja.

- Gali būti sprendžiama skaidant segmentą jo viduje (puslapiuoti segmentai)

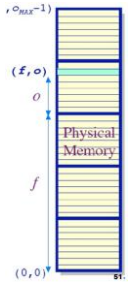
4.18 Puslapiavimo principai (adreso sudedamosios dalys), adreso transliacijos schema. Su puslapiavimu susijusios problemos

Tai viena iš pagrindinių atmintinės paskirstymo strategijų.

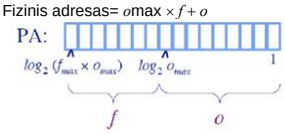
Fizinė atmintis yra sudaloma į fiksuoto, puslapio dydžio blokus, vadinamus rėmais (frames).

Atminties adresas yra rinkinys (f, *o*):

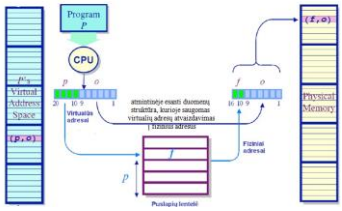
f- rėmo numeris (f*max* rėmų)
o –poslinkis nuo rėmo pradžios (*omax* baitai/rėmai)
Fizinis adresas= $o\text{max} \times f + o$



PA:



Virtualaus adreso transliavimas į fizinį



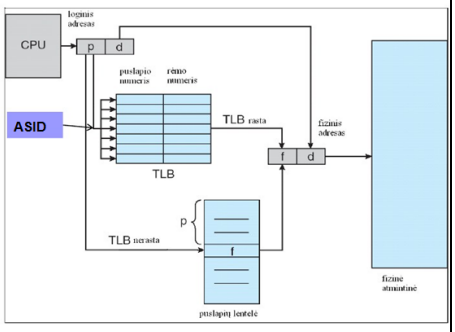
Du kreipiniai į atmintį; - TLB lentelės

Puslapio dydis - 4KB, 8KB arba 16 KB

Rezidentinis proceso dydis - Vienodi rėmų kiekiai, Rėmų kiekis proporcingas procesų dydžiui

4.19 Adreso transliavimo schema naudojant TLB įrašus

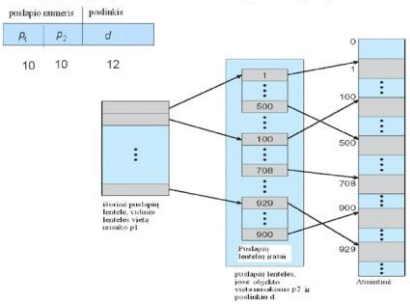
Adreso transliavimas naudojant TLB įrašus



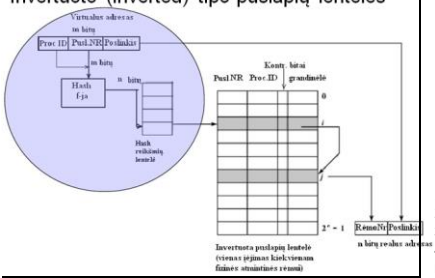
4.20 Puslapių lentelių tipai, jų naudojimo principai transliuojant adresą.

- Vieno lygio,
- hierarchinės,
- invertuotos santraukos (hash) tipo, puslapių lentelės

Dviejų lygių hierarchinė puslapių lentelė



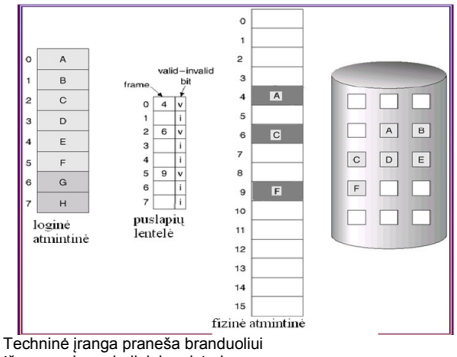
Invertuoto (inverted) tipo puslapių lentelės



4.21 Puslapio trūkumo situacija ir jo apdorojimo schema

OS užsiima puslapiavimu 4 atvejais
1. Proceso sukūrimas
nustatomas programos dydis
Sukuriama puslapių lentelė
2. Proceso vykdymas
MMU (atmintinės valdymo įrenginys gauna informaciją apie naują procesą)
TLB (translation lookaside buffer) buferis išvalomas
3. Esant puslapio trūkumui
Nustatomas virtualus adresas, sukėlęs puslapio trūkumą
Iškeliamas kažkuris puslapis į diską ir reikiamas puslapis įkeliamas
4. Procesui baigus darbą
Atlaisvinama puslapių lentelė, puslapių remain

Puslapio trūkumas (page fault)



1. Techninė įranga praneša branduoliui
2. Išsaugomi pagrindiniai registrai
3. OS nusprendžia, koks virtualus puslapis yra reikalingas
4. OS patikrina ar adresai yra teisėti, ieško tinkamo pagr. atmintinės rėmo.
5. Jei parinktas rėmas yra modifikuotas (dirty), jis perrašomas į diską.
6. OS įkelia naują puslapį iš disko
7. Atnaujinamos puslapių lentelės

Grįžtama prie komandos sukėlusios puslapio trūkumą:

- Planuojamas proceso tęsimas
- Atstatomi registrai
- Programa tęsia savo veiksmus

4.22 Rezidentinis puslapio dydis, jį apibrėžiantys veiksniai, puslapių mainų įtaka jo dydžiui.

Rezidentinio kiekio minimumo bei maksimumo reikšmės.

- Maksimalų kiekį riboja aktyvių procesų kiekis
- Maksimalų kiekį riboja laisvų rėmų kiekis.

Priskyrus mažai puslapių gali smarkiai išaugti puslapių mainai.

4.23 Puslapių keitimo algoritmai, kiekvieno jų pliusai, minusai

Reikia tam tikro algoritmo.
Funkcionalumas – reikia tokio algoritmo, kuris garantuotų mažiausią puslapių trūkumų kiekį.
Kai kurie proceso puslapiai gali būti kelis kartus įkeliami į

4.24 Bendrai naudojamos atmintinės sritys

Procesai dažnai naudojasi tom pačiom programom, funkcijom, ...
Procesams reikia leisti šiuo kodu naudotis bendrai.
Privalumai:

- Nereikėtų laikyti daug vienodų kopijų.

Minimalų kiekį nusako architektūros naudojamos komandos:
- Turi valdyt tiek skirtingų rėmų, į kiek skirtingų rėmų gali kreiptis bet kuri komanda.
- Komanda taip pat gali būti susieta su kelias rėmais

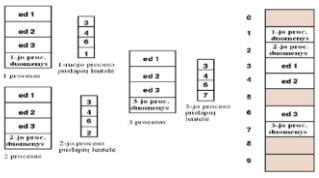
Kai laisvos pagrindinės atmintinės kiekis nukrinta žemiau leistinos ribos, operacinė sistema iškelia tuos proceso puslapius, kurie viršija
minimumo reikšmę.
Jei procesas turi mažiau puslapių nei min – procesas suspenduojamas

atmintinę.
Įvykus pertraukimui dėl puslapio trūkumo reikia parinkti puslapį kurį reikia keisti.
Optimalu būtų pakeisti tą puslapį, kurio nereikės ilgiausią laiką.
Algoritmai:
1.Keičiamas puslapis, kurio prireiks vėliausiai.
(yra optimalus, bet jis near praktiškai taikomas, nes juo yra sunku nusakyti, kokie kreipiniai ir į kurį puslapį vyks ateityje)
2.Keičiamas atsitiktinai parinktas puslapis.
(nesunku įgyvendinti, bet jis nepasižymi geru funkcionalumu)
3.Keičiamas paskutiniu metu nenaudotas puslapis (NRU). (stengiamasi atmintyje išlaikyti tuos puslapius, į kuriuos neseniai buvo kreiptasi ir pagal šį algor. yra skaitomas, kad puslapis, į kurį buvo neseniai kreiptasi, yra svarbesnis, nei modifikuotas puslapis.)
4.Puslapiai keičiami laikantis FIFO disciplinos.
(surikiuoja į eilę puslapius, įkeltus į pagrindinę atmintinę, eilės priekyje - seniausi, gale - šviežiausi puslapiai. Jį taikant near gaunami geri rezultatai, nes yra išmetami vienodu dažniu tiek tie puslapiai, kurie near dažnai naudojami, tiek tie, į kuriuos pastoviai vyksta kreipiniai)).
5.Laikrodžio (clock) algoritmas. (dar efektyvesnis, palaiko žiedinį sąrašą, kuriame laikrodžio rodyklė rodo į seniausią puslapį.)
6.Keičiamas mažiausiai paskutiniu metu naudotas puslapis LRU (Least Recently Used)
(yra tikima, kad tie puslapiai, kurie yra naudojami bus naudojami vėl netrukus).
7.Nedažnai naudoto puslapio keitimas NFU (Not frequently used)(dauguma kompiuterinių sistemų neturi techninio palaikymo tiksliai LRU algoritmo realizacijai.Šis algo. gali būti imituojamas. Puslapis su mažiausia R bitų reišme yra parenkamas keitimui.

- Tai sumažintų programų naudojamos atmintinės kiekį.

Reikalavimai:
- Šio bendrai naudojamo kodo procesai neturėtų keisti.
- Jis turėtų būti patalpintas srityje, kuri būtų žinoma visiems procesams.
- Operacinė sistema tokiu atveju turi identifikuoti puslapius kaip bendrai naudojamus arba tokius, kuriais nėra bendrai naudojamasi.
- Tokiu atveju tik kiekvieno proceso privatus kodas bei duomenys turėtų būti laikomi skirtingose atmintinės vietose.

Pavyzdys, kai trys procesai naudojasi tais pačiais puslapiais ed1, ed2, ed3.



5.25 Failai ir failų sistema (savokų apibrėžimai), failų sistemos tikslai, sprendžiami uždaviniai, failų sistemos abstrakcijos lygiai

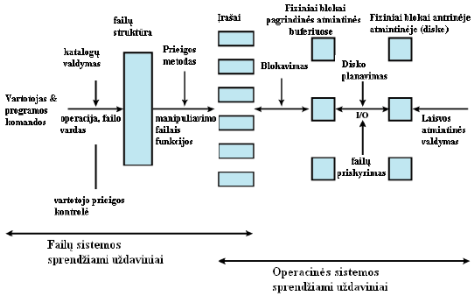
Failų sistema ir pateikia būdą kaip saugoti, pateikti ir valdyti informaciją nuolatinio saugojimo įrenginyje kokiu yra diskas.

- Failais vadinamas bet koks, turintis vardą duomenų rinkinys, kuriuo yra manipuliuojama kaip atskiru vienetu.
- Operacinė sistema gali sukurti vartotojui sąsają, kuri palengvina jam navigacijos veiksmus su failais. Šia sąsaja yra failų sistema.

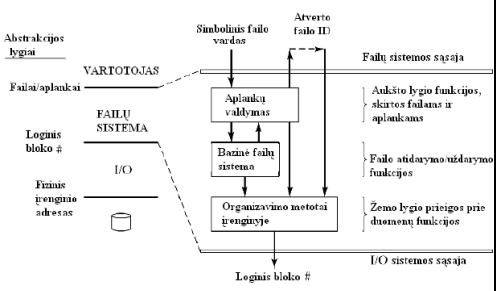
Failų sistema teikia metodą, skirtą failų saugojimui ir organizavimui
- Teikti loginį (abstraktų) požiūrį į failus ir katalogus
- Užtikrina loginę failų organizaciją,
- Padėti efektyviai naudoti atmintinės įrenginius
- Palaikyti bendrą naudojimąsi duomenimis

Failų sistema apima:
- diske saugomų *failų visumą*;
- *duomenų struktūras*, reikalingas failams tvarkyti (aplankai, failų deskriptoriai, informacija apie vietą diske);
- *failus tvarkančias programas* (skirtas failų kūrimui, naikinimui, kopijavimui, perkėlimui, įrašymui ir t.t.).

Failų sistemos sprendžiami uždaviniai:



Abstrakcijos lygiai:



5.26 Failą apibūdinančios charakteristikos, prieigos prie failų būdai

Failo charakteristikos:

Informacija apie failus yra laikoma aplankų struktūroje (diske). Tai:
- Failo tipas – reikalingas tik sistemoms, kurios palaiko skirtingus failų tipus
- Failo savininkas,
- Šie duomenys naudojami apskaitai, prieigos prie failų nustatymui
- Leidimai prieigai prie failo,
- Nusako kas ir kokius veiksmus gali atlikti su failu
- Data, laikas
- Failo sukūrimo, modifikavimo, prieigos data
- Failo vieta diske
- Failo dydis ir kitos

Prieigos prie failo būdai:

Failų sistemos gali teikti skirtingus prieigos prie failų metodus, tai:
- Nuosekli prieiga
- Tiesioginė prieiga
- Atsitiktinė
- Asociatyvi

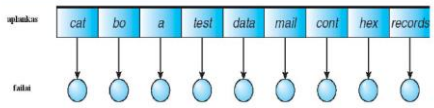
- **Nuosekli prieiga:**
- Visi baitai/įrašai skaitomi nuosekliai nuo pradžios
- Negalima šokti atgal skaitymo metu (reikia persukti juostą)
- Yra taikoma magnetinių juostų atveju
- **Atsitiktinis išrinkimas:**
- Baitai arba įrašai skaitomi bet kuria tvarka
- Tai svarbu duomenų bazių sistemose

5.27 Aplankų struktūros, siekiami tikslai, jų įrašuose saugoma informacija

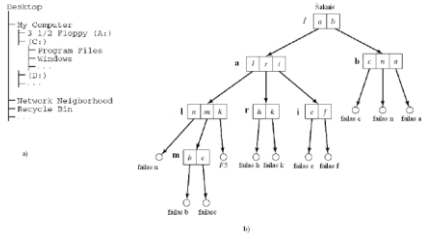
Pats aplankas taip pat yra failas, tik su specialia struktūra.
- Kiekvienas įėjimas aplanke – tai simbolinis failo vardas kartu su nuoroda į tai,
kur failas yra patalpintas diske.
- Be to dar gali būti saugoma informacija apie failo dydį, tipą, prieigos galimybes,
modifikavimo ir sukūrimo laikai.

Aplankų struktūros:

- Dauguma failų sistemų palaiko daugelio lygių aplankų struktūras.
- Dažniausiai naudojamos medžio tipo struktūros , nes naudojant vieno lygio struktūrą atsirastų problema įvardinant failus (negalima būtų naudoti tų pačių vardų), negalima būtų panaudoti grupavimo.



Hierarchinės aplankų struktūros



Aplankų įrašuose saugoma informacija:

Galima saugoti tik simbolinius failų vardus ir metaduomenis
Galima saugoti tik simbolinius failų vardus ir nuorodą į aprašą.
- Tokiu atveju reikia papildomo kreipinio į diską, norint paimti informaciją, esančią failo deskriptoriuje.

Įėjimų į aplanką saugojimui irgi gali būti parenkami įvairūs variantai:
- Tai gali būti ir fiksuoto ilgio masyvai,
- surišti sąrašai,
- „hash“ lentelės
- B medžiai.

Bazinė failų sistema, vykdydama failų atidarymo/ uždarymo veiksmus paima arba nustato failą aprašančią informaciją, naudojamą efektyviai prieigai prie failo.

5.28 Bazinės failų sistemos sprendžiami uždaviniai

5.29 Failų fizinio organizavimo būdai, kiekvieno iš jų

5.30 Laisvos disko vietos valdymo būdai

Bazinė failų sistema, vykdydama failų atidarymo/ uždarymo veiksmus paima arba nustato failą aprašančią informaciją, naudojamą efektyviai prieigai prie failo.

Bazinė failų sistema:
Aktyvuoja arba deaktyvuoja failus juos atverdama arba užverdama

- Vykdo failų atidarymo/uždarymo (open/close) veiksmus
- Jei reikia vykdo vartotojo prieigos teisių tikrinimą
- Paima arba nustato failą aprašančią informaciją, kuri naudojama prieigai prie failo.

Failo aprašas (i-node Unix)

- savininko id
- Failo tipas
- Apsaugos informacija
- Atvaizdavimas į fizinius disko blokus
- Sukūrimo laikas, paskutinio panaudojimo laikas, paskutinės modifikacijos laikas
- Kreipinių (ryšių)kiekis

pliusai, minusai

Failų fizinė organizacija:

- Fizinė failų organizacija aprašo failo išdėstymo išorinėje atmintyje, pvz., diske, taisykles.
- Failas susideda iš fizinių įrašų – blokų.
- Blokas – tai mažiausias duomenų vienetas, kuriuo išorinis įrenginys apsikeičia su pagrindine atmintine.
- Priskiriant disko blokus failams yra siekiama efektyviai išnaudoti disko erdvę.

Failų organizavimo būdai:

Problema – paskirti failams vietą diske: Failai – tai blokų eilė

- Disko erdvė turi būti išnaudota efektyviai
- Failus turi būti lengva išrinkti iš disko

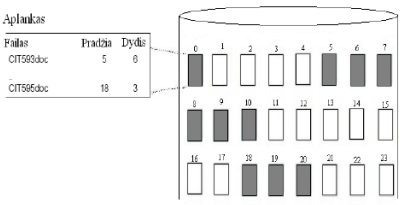
Naudojamos talpinimo strategijos:

- Į nuosekliai einančius blokus.
- Į blokus, sujungiant juos į sąrašą.
- Į blokus, formuojant indeksų sąrašą

Failų organizavimo būdai:
Nuoseklus talpinimas

Failas diske talpinamas į nuosekliai einančius disko blokus. Šis talpinimo metodas reikalauja failams priskirti tiek nuosekliai einančių blokų, kiek reikalauja to failo dydis

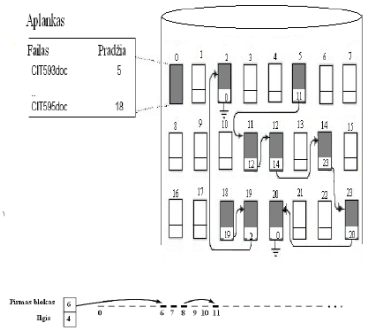
- Paprastai įgyvendinamas
- Greita **nuosekli** prieiga (minimalus galvutės judesys)
- Sunku atlikt įterpimą/išmetimą
- Sunku nuspręsti, kiek vietos priskirti pradžioje
- Išorinė fragmentacija galima



surištas sąrašas

Šiuo atveju aplanke yra nurodoma failo pradžios blokas, o jau tame bloke patalpinama nuoroda į sekantį failo bloką

- Paprasta įterpti/išmesti, nėra išorinės fragmentacijos
- Nuoseklus išrinkimas mažiau efektyvus (paieška, vėlinimas)
- Nėra galimas tiesioginis kažkurio bloko išrinkimas
- Mažas patikimumas (suirus grandinėlei)



indeksinė organizacija:

- suformuojama indeksų lentelė, rodanti, kuriame fiziniame bloke yra patalpintas atskiras failo blokas .
- Paprasčiausiu atveju ši indeksų lentelė gali būti laikoma failo apraše.
- Esant tokiam talpinimui paprasta atlikti įterpimo, išmetimo veiksmus, galimas tiek nuoseklus, tiek tiesioginis failo blokų išrinkimas.
- Trūkumas pasireiškia tame, kad failo dydžius gali riboti indeksų lentelės dydžiai.

Panašios problemos kaip ir valdant pagrindinę atmintinę:
Saugoma informacija apie laisvus blokus surišto sąrašo metodu

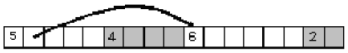
- Surišti individualius blokus mažai efektyvu :
  Nėra taikomas blokų apjungimas kad būtų mažinamas paieškos laikas
- Surišamos nuoseklių blokų grupės
  Grupė blokų yra priskiriama arba atlaisvinama vienu metu

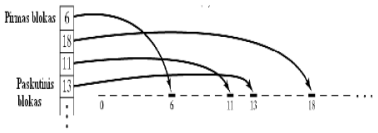
Dvejetainio žemėlapio formavimas (bitmap)

- Analogiškai kaip ir pagrindinės atmintinės atvejui.
- Trūkumas dvejetainio žemėlapio yra tame, kad ieškant laisvo bloko failų sistemai gali tekti peržiūrėti visą dvejetainį žemėlapį
- Reliatyviai paprastas metodas – lengva rasti n laisvų gretimų blokų
- Naudoja :Intel 80386, Motorolla 68020/30, Apple Macintosh
- Nėra efektyvu, jei visas žemėlapis nėra laikomas pagrindinėje atmintinėje
- Sunku naudoti, jei diskai yra labai didelės talpos

--|--

- Nėra tikslinga surišti į sąrašą visus atskirus laisvus blokus po vieną.
- Jei yra keli iš eilės einantys laisvi blokai, tai jie galės būt ir priskiriami esant poreikiui, todėl laisvų blokų sąrašas ir formuojamas surišant į sąrašą laisvų blokų grupes
- Laisvi blokai yra priskiriami failams imant juos iš sąrašo pradžios, o atsilaisvinę blokai yra įjungiami į šio sąrašo pabaigą.





Indeksavimo variantai:

Kelių lygių indeksų hierarchija

- pirmo lygio indeksai rodo ne tiesiogiai į disko blokus, o į antro lygio indeksus
- problema: auga kreipinių į diską kiekis augant hierarchijos lygių kiekiui

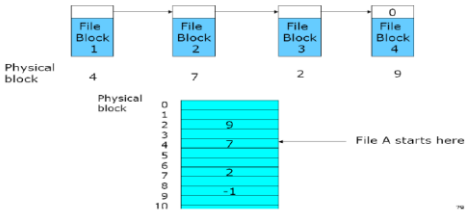
Priauginimo tipo indeksacija

- dalis indeksų rodo tiesiogiai į disko blokus, o jei jų nepakanka, yra įtraukiami papildomi indeksų lygiai, panaudojama dviguba, triguba indeksacija

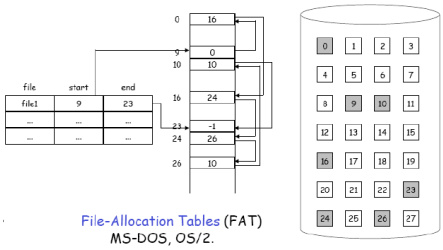
Tarkim jei indekso bloke galim turėt 256 nuorodas, tai bus 65536 nuorodų esant dviejų lygių indeksams, 1K blokas, 64M failas

5.31 Windows failų sistemos: FAT, NTFS

Įrašus apie talpinimo vietą deda į failų talpinimo lentelę, vadinamą FAT (File Allocation Table).



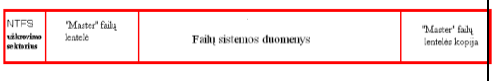
FAT



File-Allocation Tables (FAT)
MS-DOS, OS/2.

- FAT privalumai yra tame, kad informacija apie disko blokų užimtumą yra saugoma nedidelėje diske esančioje lentelėje.
- Lentelė indeksuojama blokų numeriais – vienas įėjimas-vienas blokas.
- Užimti blokai tarpusavyje surišami sąrašu
- Neužimti blokai indikuojami 0-nės reikšmės įėjimu.
- Patogus yra laisvų blokų suradimas ir priskyrimas failui – šiuo tikslu pakanka surasti neužimtus FAT lentelės įrašus.
- Kiek sudėtingesnis yra nuoseklus failo blokų skaitymas.
- naudoja MS-DOS ir OS/2
- Kiekviena FAT versija naudoja skirtingo ilgio įrašus. Įrašo ilgis yra atspindimas ir FAT versijos vardu – FAT16 lentelės atveju naudojami 16 bitų įrašai, FAT 32 – 32 bitų.
- FAT16 sistemos atveju failo maksimalus dydis gali siekti 2GB,
- FAT32 iki 4TB

NTFS (New Technology Filesystem):



• NTFS failų sistema kiekvieną failą (arba katalogą) traktuoja kaip ***failo atributų rinkinį***.
• Atributais g.b. tokie elementai kaip failo vardas, informacija susijusi su failo apsauga, ir net patys failo duomenys.
• Kiekvienas atributas identifikuojamas jo kodu (arba vardu).
• Kai failo atributai telpa į MFT failo įrašą, jie vadinami rezidentiniais, pvz failo vardas, laiko atžyma (time stamp).
• Atributai, kurie netelpa į įrašą yra nerezidentiniai ir jie talpinami viename ar keliuose disko "klasteriuose"
• NTFS sukuria atributą List aprašymui, kur yra patalpinti visi failo atributų įrašai.

5.32 Žurnalinės failų sistemos, jose taikomi principai

Žurnalinė failų sistema (JFS):

- Operacijos, apie kurias rašomi įrašai:
  – create, link, mkdir, truncate, allocating write, …
- Kiekviena sistemos vykdoma operacija gali būti surišta su eilės metaduomenų keitimais (naujinimais).
- Realūs įrašymai į diską gali būti vykdomi :
  – asinchroniškai,
  – prieš darant įrašą

Žurnalinės failų sistemos

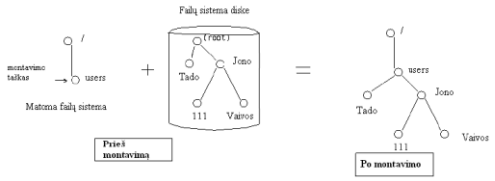
- Privalumai:
  – Asinchroninis metaduomenų rašymas
  – Greitas atstatymas: jis priklauso tik nuo žurnalo dydžio o ne nuo failų sistemos dydžio
- Trūkumai
  – Papildomi įrašymai į diską
  – Reikia vietos žurnalų saugojimui (nereikšminga)
- Dauguma modernių Linux failų sistemų naudoja žurnalines failų sistemas.

5.33 Failų sistemos montavimas, atsarginės kopijos

Failų sistemos montavimas UNIX

- Kiekviena diske esanti failų sistema turi savo šakninį katalogą
- Galima sukurti bendrą failų sistemą ją sumontuojant iš atskirų failų sistemų
  – Tai yra sukurti ryšį tarp katalogo vienoje failų sistemoje ir šakninio katalogo kitoje failų sistemoje.
- Tai leidžia sukurti bendrą medį iš kelių failų sistemų.
- Tai gali būti išplečiama ir per tinklą, sumontuojant kelių kompiuterių failų sistemas.

- Atskirame diske esančios failų sistemos įjungimas į egzistuojančią katalogų struktūrą yra žinomas kaip failų sistemos montavimas
- Primontuojama failų sistema gali rastis viename iš tiesiogiai prijungtų diskų (lokali failų sistema) arba būti nutolusios tinklinės failų sistemos dalimi.
- Montavimas susieja primontuojamą failų sistemą su tam tikru egzistuojančios failų sistemos katalogu.
- Iki montavimo failai, esantys primontuojamame diske nėra pasiekiami vartotojams.
- Katalogas, kuriame vykdomas montavimas yra vadinamas montavimo tašku.
- Montavimo tašku turi būti tuščias egzistuojančios sistemos katalogas.



Failų sistemos montavimas Linux

- Linux sistemos atveju yra galimybė sujungti skirtingo tipo failų sistemas į vieną bendrą failų sistemą.
  – Tai gali būti tokios failų sistemos: FAT, FAT32, NTFS, ir HPFS
- Jei šaknis katalogas viename iš diskų gali būti c:\, tai tas pats diskas Linux sistemoje gali būti pasiekiamas keliu /mnt/xdir; čia xdir tai montavimo taškas.
- Įrenginį sumontavus, jis tampa pasiekiamas vartotojams, turintiems leistiną prieigą prie jo.
  – Pvz diskelio primontavimui taške /mnt/floppy, reikės komandos `mount /dev/fd0 /mnt/floppy`
- Linux instaliavimo metu, informacija apie montavimo taškus yra sukuriama ir patalpinama faile /etc/fstab

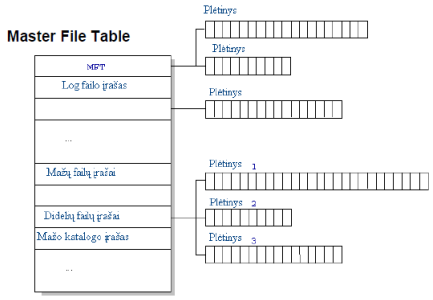
Failų sistemos išmontavimas

- Išmontavus failų sistemą ji tampa laikinai neprieinama vartotojams.
- Išmontavimas neišmeta failų sistemos iš disko
- Sumontuota failų sistema yra automatiškai išmontuojama po kompiuterio stabdymo komandos "shutdown".

Failų sistema ir įvykiai, dėl kurių prarandami duomenys:

- Techninės įrangos sukeltos problemos

NTFS struktūra



- Programinės įrangos klaidos
- Atsitiktiniai praradimai
- Priešiškai nusiteikę vartotojai ar įsilaužėliai
- Natūralios ar žmogaus sukeltos nelaimės

Atstatymas:

- Atstatyti prarastus failus atstatant duomenis iš atsarginių kopijų.
  – Leidžia sumažint duomenų praradimo riziką.
  – Yra vertinga, jei iš jų galima atstatyti duomenis
- Naudojamos sisteminės programos disko atsarginių kopijų darymui (*back up)* į kitus įrenginius.

Saugojimas –backup:

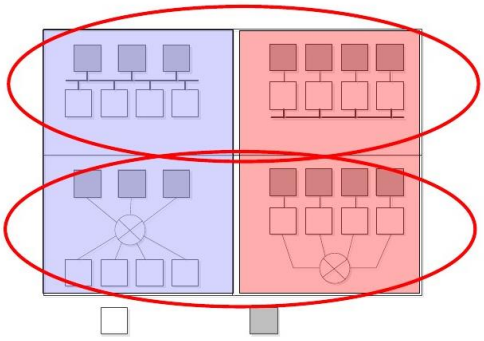
Kopijų darymas:

- Ką saugoti?
  – Operacinę sistemą – gal saugoti jos įvaizdį (image)
  – Duomenis
- Kur saugoti?
  – Atskiri diskai
  – USB flash tipo įrenginiai
  – USB išoriniai diskai
- Įvesti tam tikrą tvarką saugant duomenis, kuriuos norima atstatyti
  – /var ; c:\data
  – UNIX – vartotojų failai, /etc /usr/local /var

6.34 Mobilieji įtaisai, jų savybės. Android OS programų steko architektūra, Dalvik virtuali mašina. Apribojimai programinei įrangai

6.35 Virtualizacija, jos tipai, virtualizacijos taikymai. Debesų kompiuterija ir jos teikiamos paslaugos

6.36 Skirtingos procesorių ir atminčių organizacijos paskirstytose sistemose.



Pilkas – atmintis
Baltas – procesorius

Viršuje SMT (vienalytis modelis)
Apačioje Multicore (mišrus modelis)

6.36 UMA ir NUMA sistemos.

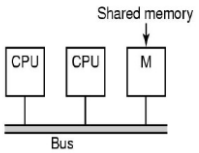
multiprocesoriaus modelyje **fizinė atmintis vienodai padalinta bendram naudojimui** tarp visų procesorių. Santrumpa UMA pabrėžia, kad bet kuris procesorius gali kreiptis į bet kurį atminties žodį, tam sugaišdamas **vienodą** laiką

multiprocesoriaus modelyje **fizinė atmintis bendra ir prieinama visiems procesoriams, bet ji padalinta**, todėl atskiri procesoriai kreipiniams į konkrečią atminties sritį sugaišta skirtingą laiką

6.37 Multiprocesorinių sistemų techninė įranga

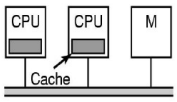
Multiprocesorių sistemų yra keli modeliai:

1) gali turėti bendrą atmintį



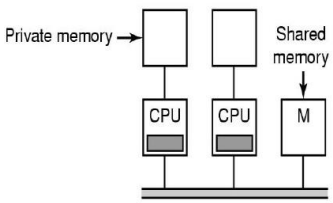
(a)

2) gali turėti bendrą atmintį ir kiekvienas savo cache



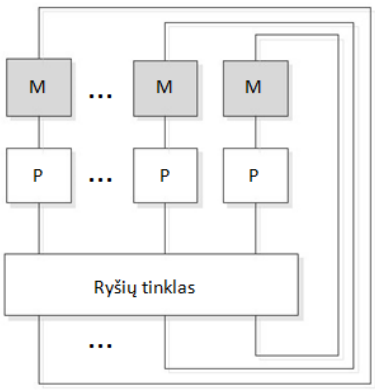
(b)

3) gali turėti bendrą atmintį, kiekvienas savo cache ir dar savo vidinę atmintį (privačią)



(c)

6.38 Paprasčiausias NUMA sistemos modelis



6.39 OS naudojimo multiprocesorinėse sistemose variantai

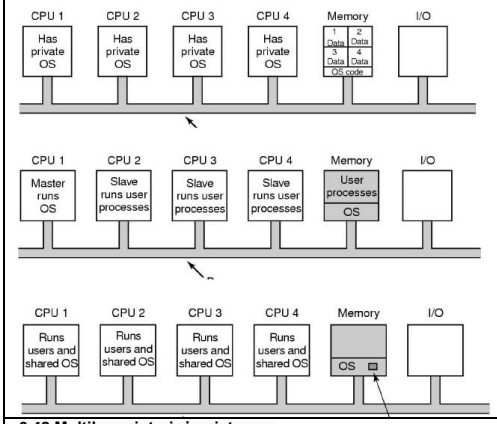
1) kiekvienas procesorius turi savo OS

6.40 Procesorių sinchronizacijos problemos sprendimai

6.41 CPU išteklių planavimas laike, erdvėje

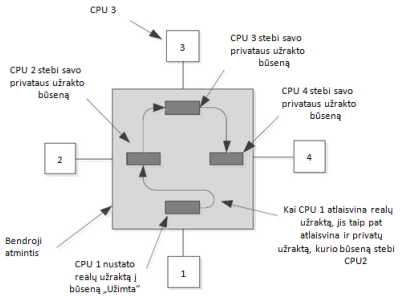
Planavimas laike (laiko skirstymas):

2) Master-Slave (asimetriniai) multiprocesoriai
3) SImetriniai multiprocesoriai



1) Magistralės blokavimas – OK, bet neefektyvu
2) Privatūs užraktai

Pvz: CPU1 nustato **realų** užraktą bendrojoje atmintyje į užimta, o like procesoriai stebi **privatų** užraktą ir nieko nedaro kol CPU1 **abiejų** užraktų neatlaisvina



1) Išmanusis planavimas (angl. smart scheduling)
2) Planavimas pagal panašumą (angl. affinity scheduling)

Planavimas erdvėje (erdvės skirstymas):
1) Išmanusis planavimas
2) Planavimas pagal panašumą
3) Labiau tinkamas – susijusiems procesams
4) Vykdant gijų I/O pertraukimo metu procesorius nėra atlaisvinamas
5) Pakoreguotas planavimo principas, jei procesai gali valdyti gijų lygiagretumo laipsnį

6.42 Multikompiuterinės sistemos

Vienalytės:

Tvirtai susieti, privačias (lokalias) atmintines turintys, vienodų parametrų procesoriai

Tipai:

1) System Area Networks (SANs)
2) Massively Parallel Processors (MPPs)
3) Clusters of Workstations (COWs)

Mišrios:

Elementai yra pilnaverčiai kompiuteriai turintys ne tik savo CPU, atmintį, tinklo sąsajas, failų sistemą, bet ir periferinius įrenginius bei visa kitą reikalingą įrangą.

Tai sistema, kurios aparatinės ar programinės įrangos komponentai išsidėstę tinkle esančiuose kompiuteriuose, komunikuoja tarpusavyje ir koordinuoja savo veiksmus apsikeisdami tik pranešimais [Colouris]

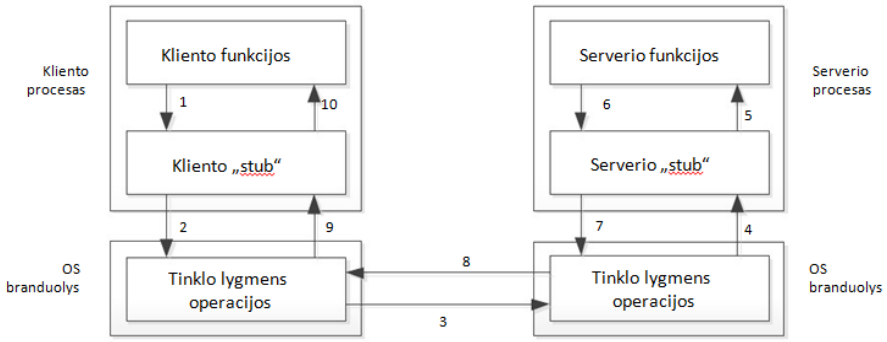
Mišrių aspektai:

Techninės įrangos aspektas, teigiantis, kad paskirstytą sistemą sudaro autonominių, vienas nuo kito nepriklausomų kompiuterių aibė.

Programinės įrangos aspektas, teigiantis, kad paskirstytos sistemos vartotojui sistema turi atrodyti kaip vientisas komponentas (skaidri).

6.43 RPC ir paskirstytoji bendra atmintinė

RPC (Remote Procedure Call) tai nutolusių procedūrų kvietinys.



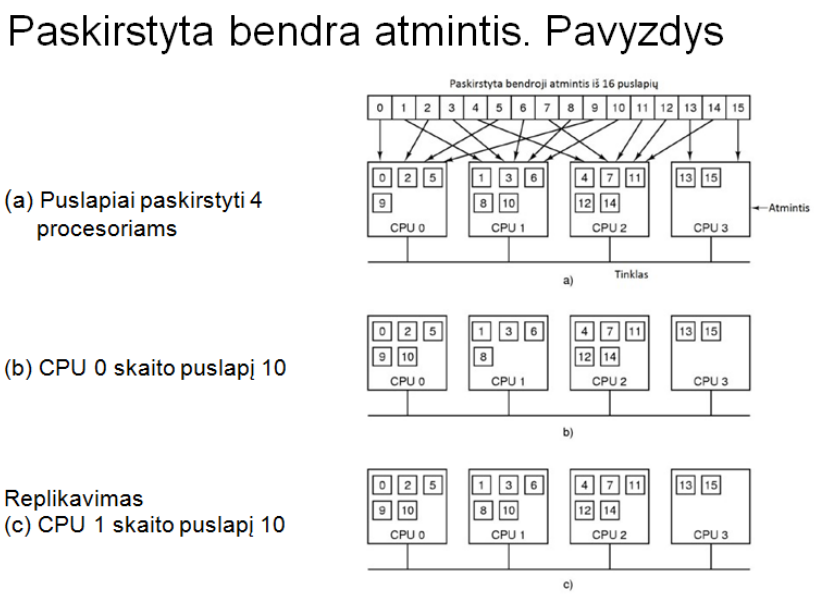
Privalumai:

1) Užtikrinama galimybė naudoti įprastą procedūrų kreipinių semantiką;
2) RPC visus žemo lygmens instrukcijas ir funkcijas paslepia po stub funkcijomis. Galima nebesirūpinti detalėmis (tokiomis kaip soketai, jungčių numeriai, baitų tvarka (I/O atvejis)

Trūkumai:

1) Problematiškas rodyklių/adresų perdavimas
2) Negalimas globalių kintamųjų naudojimas

Paskirstyta bendra atmintis:



6.44 Paskirtytų failų TPĮ

TPĮ – tarpinė įranga

Charakteristikos:

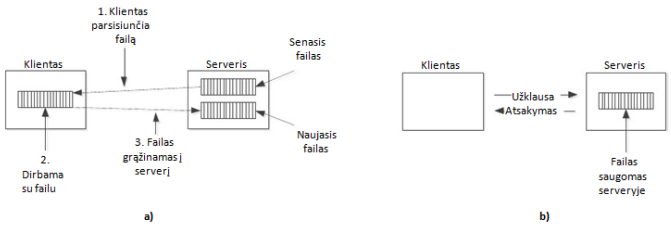
1) kirstytos failų sistemos;
2) RPC mechanizmas;
3) Paskirstyti objektai;
4) Paskirstyti dokumentai.

Tai yra modelis kai laikoma, kad viskas yra failas;
TPĮ užtikrina vietos skaidrumą (angl. Location transparency) tik paprastiems duomenų saugojimui skirtiems failams;
Realizacinis pavyzdys – paskirstyta failų sistema.

Svarbūs aspektai kuriant tokią įrangą:

1) Failų persiuntimo modelio parinkimas;
2) Aplankų hierarchijos formavimo principo parinkimas;
3) Vardų vientisumo/skaidrumo įgyvendinimas;
4) Dalinimosi failais semantikos parinkimas.

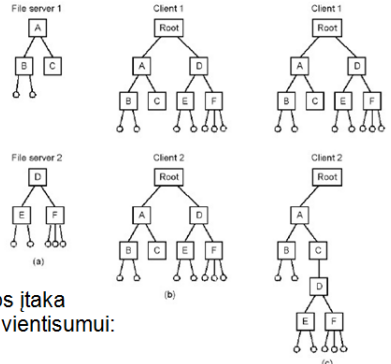
Failų persiuntimo modeliai



(a) išsiuntimo-parsiuntimo modelis (angl. upload/download)

(b) nuotolinės prieigos modelis (angl. remote access)

Aplankų hierarchijos formavimo principai bei vardų vientisumo įgyvendinimas



Failų hierarchijos įtaka vardų sistemos vientisumui:

(b) Klientai turi tą patį hierarchijos vaizdą

(c) Klientai turi skirtingą hierarchijos vaizdą

Pagrindinės trys failų dalijimosi semantikos:

1) Unix semantika.
2) Sesijos semantika.
3) Tranzakcijos semantika.

Paskirstytų OS palyginimas

Aspektas	DOS		NOS	TPĮ
	Multiproc.	Vienalyčiai		
Skaidrumo laipsnis	Labai aukštas	Aukštas	Žemas	Aukštas
Vienoda OS mazguose	Taip	Taip	Ne	Ne
OS kopijų skaičius	1	N	N	N
Komunikavimo pagrindas	Bendra atmintis	Pranešimai	Failai	Priklauso nuo modelio
Resursų valdymas	Globalus, centralizuotas	Globalus, paskirstytas	Mazgui	Mazgui
Išplečiamumas	Ne	Vidutinis	Taip	Varijuoja
Atvirumas	Uždara	Uždara	Atvira	Atvira