* 1. OS Koncepcija. (1T)

Speciali [programinė įranga](https://lt.wikipedia.org/wiki/Programin%C4%97_%C4%AFranga), užtikrinanti vartotojo sąsają ir kompiuterio [techninės įrangos](https://lt.wikipedia.org/wiki/Technin%C4%97_%C4%AFranga),[taikomųjų programų](https://lt.wikipedia.org/wiki/Taikomoji_programin%C4%97_%C4%AFranga) bei [duomenų valdymą](https://lt.wikipedia.org/wiki/Duomenys). Moderniausios operacinės sistemos sudaro galimybę dirbti daugeliui vartotojų vienu metu daugialypėje aplinkoje, užtikrina bylų (failų) apsaugą, turi daug kitų naudingų savybių. Dauguma operacinių sistemų yra pirma programinė įranga, kurią pradeda vykdyti įjungtas kompiuteris.

* 1. Kompiuterio techninę įranga
* Procesorius – Veiksmų kontrolės, duomenų apdorojimo funkcijas,

Jis nulemia kompiuterio darbo spartą. Vygdo skaičiavimo operacijas, logines operacijas. Taip pat atlieka operacijas susijusias su skaičiavimo operacijos valdymo operacijos, duomenų mainų su pagr. Ir išor. Atmintini. Taip pat Skaičiuoja iš pagr. Atminties ir perduoda į atmintinę magistralę.

* Atminties įtaisai – Jis skirtas teik vykdomų programų kodams, tiek paskirties duomenims saugoti.

Dviejų tipų pragrindinę atmintinę:

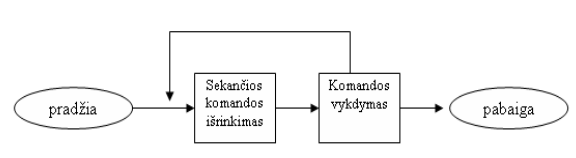
Tiesioginia kreipiantis į RAM atmintinę ir OA (operacine atmintis)

* Įvesties/išvesties įtaisai – Įvesties/išvesties operacijis vygdomos su visais išoriniais įtaisais ir tai liedžia keisti duomenis su komoiuteriu.
* Magistralė – Tai tam tikros struktūros ir mechanizmai, užtikrinantys ryšius tarp procesoriaus, ir pagrindinės atminties ir įvesties/išvesties įtaisų. Magistralė jungianti pagrindinius kompiuterio komponentus vadinama Sistemine Magistrale.
* Atminties Hierarchija:

Vidinė -> Registrai, Sparc.Atmintis, Parg.Atmintis.

Išorinė -> Magn.Diskai.

Automatinė -> Magn.Juosta.

* 1. Bazinis Prog. Vydymo ciklas
  2. Pertrauktys
* Programinės pertrauktys – Jos yra generuojamos komandų vykdymo metu, kai susidaro tam tikros situacijos:

1.Perpildymas

2.Dalyba iš nulio

3.Nekorektiška komanda

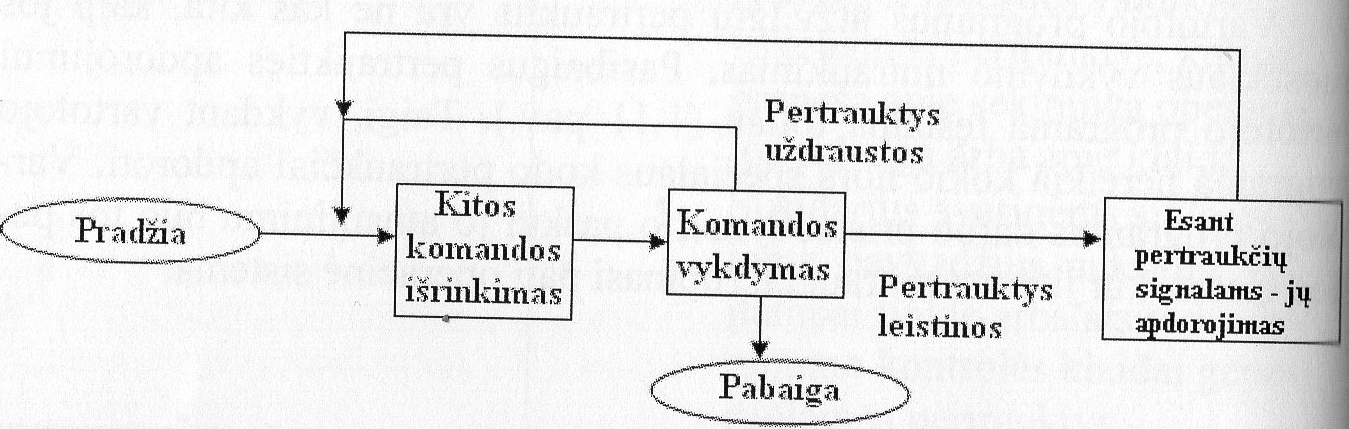
4.Kreipimasis į vartotojiu uždrausta atminties sritį.

* Aparatinės pertrauktys:

1.Pertrauktys pagal laika, generuojamos procesoriaus mechanizmo.

2. Įvesties/Išvesties sukeliamos pertrauktys, kurios praneša apie pabaigta operacija arba klaida.

3.Avarinės situacijos pertrauktys – Tinklo arba atsiradus klaidai.

* Pertraukčių apdorojimo logika:

2.1 OS Paskirtis ir Funkcijos

**OS Paskirtis**:

Palengvinti taikomųjų vartotojo programų ryšį su technine komiuterine įranga. OS naudojama kaip vartotojo ir įrangos tarpusavio sąsaja.

**OS Funkcijos**:

**Patogumas** – Naudojimasis kompiuteriu tampa paprastas ir patogus.

**Efektyvumas** – OS Sistema leidžia efektyviai naudoti kompiuterio sistemos išteklius.

**Plėtros galimybės** – OS projektuojama taip ,kad leistų efektyviai kurta, testuoti ir diegti naujas taikomąsias ir sistemines funkcijas, bet kad nekliudytų funkcijionuoti kompiuterio sistemai.

2.2 Daugelio užduočių vygdymas

Tai situacija, kai vienu metu į pagrindinę atmintį yra įkeltos ir vykdomos kelios užduotys, procesiu perjungiant nuo vienos užduoties vykdymo prie kitos.

2.3 OS Tobulinimo poreikiai

OS tobulėdamos turi prisitaikyti prie naujų paslaugų, pavyzdžiui, kad kompiuteriai pradėtų teikti internet paslaugas, tai yra palaikytis procesus, veikiančius internet tinkle, atitinkamai turėjo pasikeisti operacines sistemos.

2.4 OS Vystymosi istorija

2.5 Savybes jų vystymosi etapais

-

**Pirmasis periodas - 1945-1955 m.**

Pirmosios mašinos. Specialistai-universalai: ir projektuoja, ir eksploatuoja, ir programuoja. OS užuomazgos: jos tebuvo matematinių ir tarnybinių **paprogramių biblioteka**.

**Antrasis periodas - 1955-1965 m.**

Išaugo kompiuterių patikimumas (puslaidininkinės schemos), pasirodė pirmosios algoritminės kalbos. Sukurtos pirmosios **paketinio apdorojimo sistemos**.

Sukurtos užduočių valdymo kalbos.

**Trečiasis periodas - 1965-1980 m.**

Integrinės mikroschemos. Programiškai suderintų mašinų šeimos (IBM/360). Reikalingas ir OS suderinamumas.

OS/360 - sistema - monstras: tūkstančiai programuotojų, milijonai asemblerio eilučių, tūkstančiai klaidų.

**Multiprogramavimas**

**Laiko paskirstymo sistemos**.

**Ketvirtasis periodas - 1980 m- .**

Aukšto integracijos lygio mikroschemos. Personalinių (asmeninių) kompiuterių era.

**MS-DOS - Windows**

**Unix**

**Tinklo operacinės sistemos.**

2.6 OS Komponentai

* **Procesų valdymas** – Operacine sistema yra atsakinga už šiuos veiksmus, susijusius su vartotojų ir sistemų precesų valdymu:

1. Procesų sukūrimą ir užbaigima
2. Procesų pristabdymą ir užbaigimą
3. Techninės įrangos išteklių skirtymą procesams.
4. Procesams sinchronizuoti
5. Proceso tarpusavio ryšiams užtikrinti
6. Mirties taško situacijos išvengti

* **Atminties valdymas** – Operacinė sistema yra atsakinga už tokius uždavinius , susijusius su pagrindinės atminties valdymu.

1. Ji stebi, kurios pagrindinės atminties sritys yra užimtos ir kas jas užima.
2. Spendžia, kurį procesą įkelti į atmintį, kai ši tampa prieinama.
3. Skiria atmintinės sritį procesams arba atima ją iš proceso iškilus reikalavimiu.

* **I/O sistemos valdymas** – Sistema sudaro: patys įtaisai, buferiai, sparčioji atmintis, tvarkyklės. Kiekviemą įtaisą valdo jam skirta tavrkyklė ir tik tvarkyklei yra žinomi atskirų įtaisų savitumai. OS rūpinasi ,kad visi I/O įtaisai būtų traktuojami veinodai.
* **Failų valdymas** – Operacinė sistema atsakinga už šių veiksmų, susijusių su failaisvykdymą:

**1**.Failų sukūrimą ir šalinimą

**2**.Aplankų sukūrimą ir nakinimą

**3**. Failų įrašymą į antrinę atmintį

**4**.Failų kilnojimą tarp pagrindinės ir antrinės atminties.

* **Apsaugos sistema** – Operacinės sistemos turi įvairių apsaugos mechanizmų, kuriais turi būti kontroliuojama prieiga prie failų, procesoriaus, atmintės sričių ir kitų kompiuterių sistemos išteklių.

Tai garantuoj, kad prieiga prie šių objektų suteikiama tik autorizuotiems procesams.

* **Darbas tinkle** – Operacine sistema tokiu atveju turi garantuoti darbą tinkle. Šiam tikslui ji turi įdiegtą tinklinių protokolų dėklą ir tinklo įtaisų tvarkykles. Darbas tinkle leidžia skirtingiems, dažnai net nehomogeniniams kompiuteriams komunikuoti ir veikti tarsi didžiulei virtualiai sistemai. Pačios komunikacijos tinkle vyksta pagal tam tinkamus protokolus.
* **OS**.Paslaugos:

**1**.Programų kūrimas

**2**.Programų valdymas

**3**.I/O Veiksmai

**4**.Su failų sistema susija veiksmai

**5**.Komunikacijos

**6**.Klaidų aptikimas

**7**.Papildomos funkcijos

**8**.Išteklių priskyrimas

**9**.Apskaita

**10**.Apsauga

**2**.7 Sistemos kveitimai

* Sisteminiai kvietiniai teikia sąsają tarp taikomųjų procesų ir operacinės sistemos.
* Sisteminiai kvietiniai leidţia procesams uţprašyti OS tam tikrų paslaugų, kurių procesams patiems neleidţiama vykdyti.
* Šie veiksmai daţnai yra susiję su privilegijuotų komandų vykdymu.
  1. OS architektūros sąvoka.

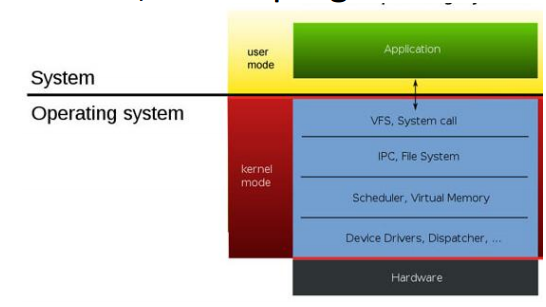
3.2 OS arch. Veikimo principai PRO‘S & CON’S

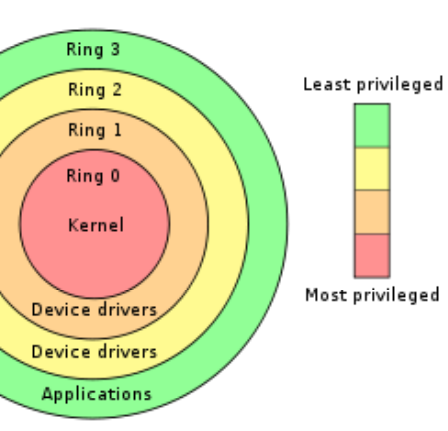
Tradiciškai OS skirstomos pagal branduolio struktūrą:

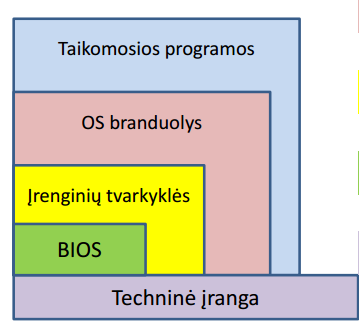
* Monolitinės turi vientisą, visas funkcijas atliekantį branduolį, kurį, keičiant aparatūrą, reikia perkompiliuoti.
* Generatyvinės turi branduolį, veikiantį, kaip monolitinis, tačiau branduolys generuojamas jo pakrovimo metu.
* Modulinės artimos generatyvinėms, tačiau turi branduolius, susidedančius iš modulių, kuriuos galima keisti OS darbo metu.
* [Mikrobranduolius](https://lt.wikipedia.org/w/index.php?title=Mikrobranduolys&action=edit&redlink=1) turinčios veikia, kaip minimalūs branduoliai, atiduodantys OS funkcijas įprastiems procesams.

Taip pat, OS neretai skirstomos pagal daugelį kitų parametrų, pvz., procesų valdymo metodus (pranešimų eile ar procesų perjungimu pagrįstos), aparatūros abstrakciją ([virtualizuojančios](https://lt.wikipedia.org/w/index.php?title=Virtuali_ma%C5%A1ina&action=edit&redlink=1) ir nevirtualizuojančios), sistemines funkcijas (pvz., [Monitorius (OS)](https://lt.wikipedia.org/wiki/Monitorius_(OS))), galimybes dirbti tinkle, istoriškai susiklosčiusias savybes (pvz., [UNIX](https://lt.wikipedia.org/wiki/UNIX), [Windows](https://lt.wikipedia.org/wiki/Windows)) ir pan.

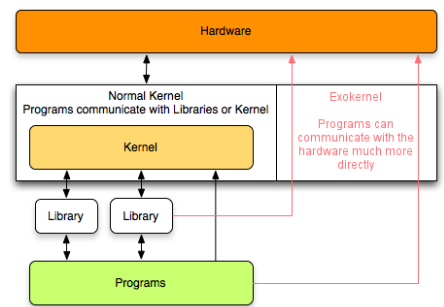
* 1. OS arch. Pavyzdžiai

 Momolitines Pvz:

Keleto lygmenų Pvz:



Virtualieji komp. Pvz:

Kitos arch Pvz:

4.1 Proceso savoka/apibrėžimas. Proceso komponentai

Procesas – tai programa jos vykdymo metu. – Programa – tai statinis failas (pasyvus elementas) – Procesas = vykdoma programa = programa + vykdymo būvis (aktyvus elementas).

pagrindiniai komponentai:

– Proceso adresų erdvė

• op. atminties dalis, kuri yra pasiekiama procesui

• Skirta įvairių komponentų (programos teksto, kintamųjų, steko, etc.) saugojimui

– Procesoriaus (CPU) būsena

• Su programos vykdymu susiję CPU registrai

• Bendros paskirties, PC, SP registrai

– OS priskirti kiti resursai

• Atidaryti failai, tinklo susijungimai, etc.

• Proceso valdymui skirta info (proceso id, savininkas, etc.)

• Statistinė info (pvz. kiek CPU, RAM naudoja procesas)

4.2 Pagrindinės procesų valdymą apimačios funkcijos.

1. Išteklių priskyrimą procesams.

2. Procesų planavimas: reikia garantuoti ,kad kiekvienas iš jų bus laiku ukurtas, vykdomas, užbaigtas, stebėt, ar efektyviai naudojamas procesorius, ar laiku gaunamas atsakymas.

3. Tarpusavio komunikavimo užtikrinimas,jų sinchronizacija ir mirties taško susidarymo prevenciją.

4.3 Procesas atmintyje

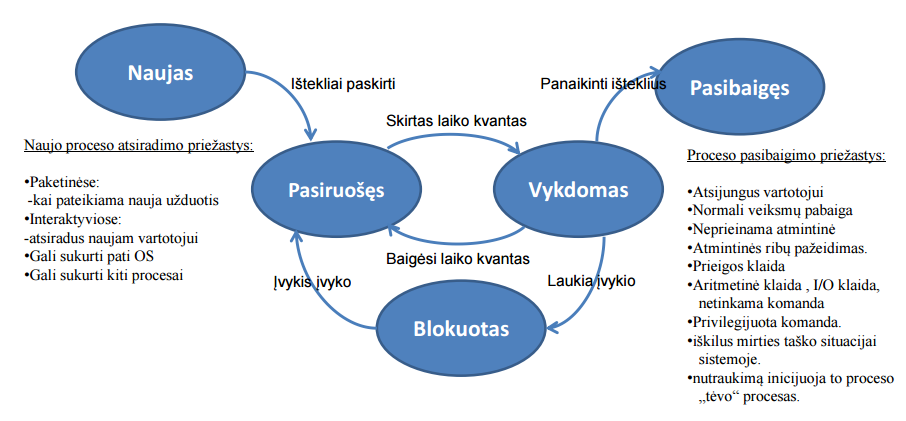
* Steko sritis (stack). Saugomi lokalių funkcijų kintamieji

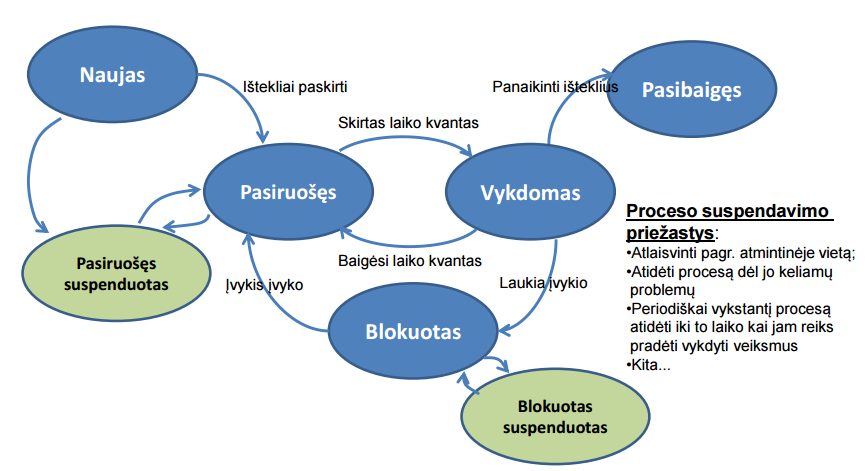
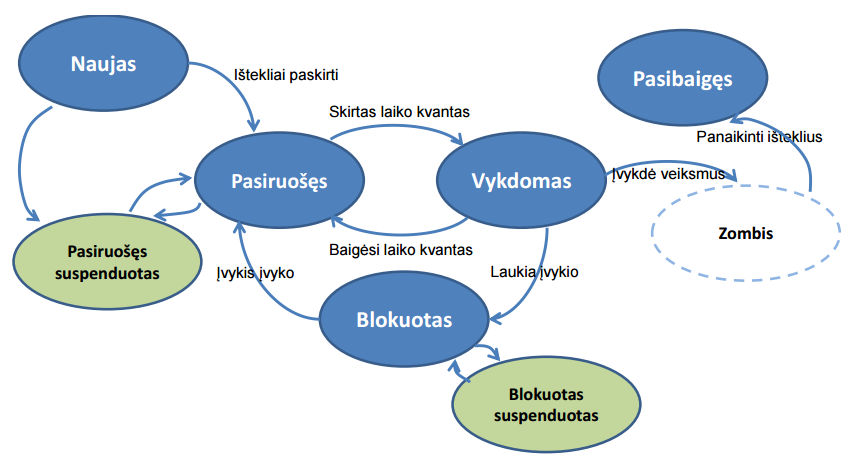
• Heap sritis. Saugomi dinamiškai kuriami objektai. (C – malloc(), C++ - new).

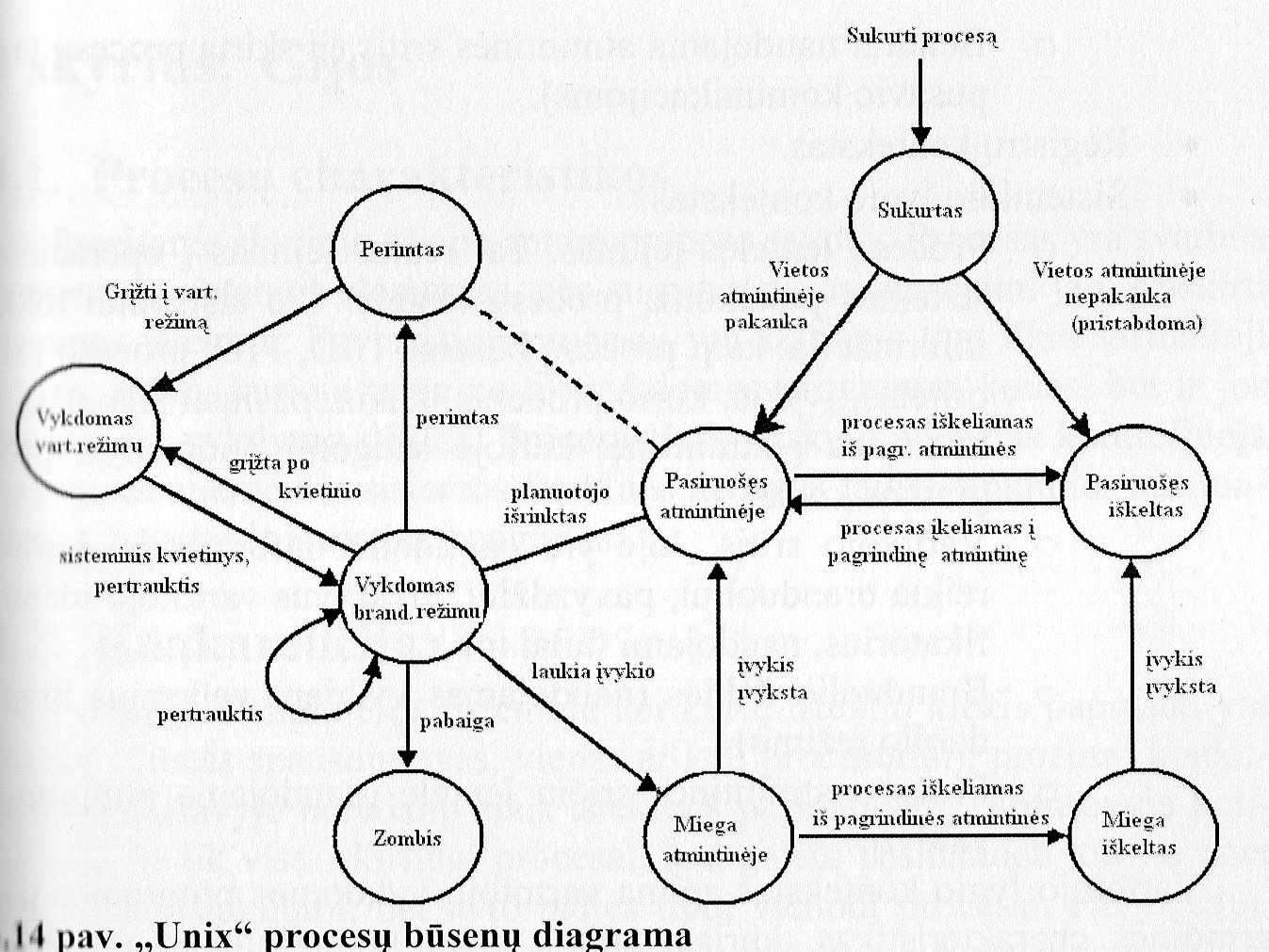
• Duomenų sritis (data). Saugomi globalūs kintamieji

• Programos kodo sritis (text). Saugomas objektinis vykdomasis programos kodas

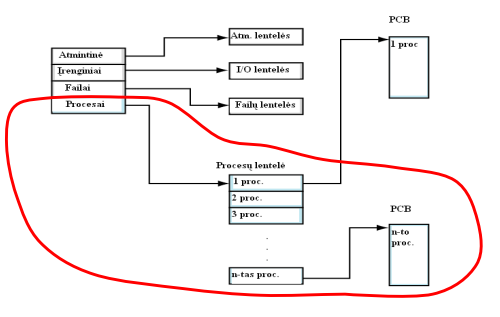
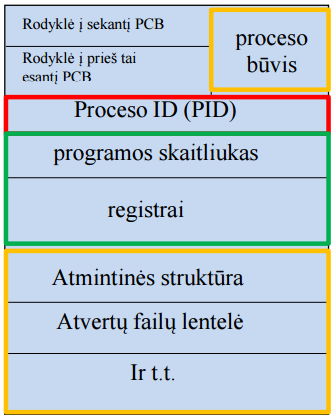
4.4 Procesų būsemų diagramos (3,5,7 busena).Tipine UNIX busenu diagrama.Zombie busena







4.5 Procesų valdymiu naudojamos duomenų struktūros.

* Procesų lentelė: OS info apie procesus saugo procesų lenteleje. 
* Procesų kontrolės blokas:

1.Proceso identifikaciniai duoemnys

2. Procesoriaus būsenos duomenys

3.Procesams vygdyti skirti duomenys.

* Procesų eilės :

struct task\_struct {

volatile long state;

void \*stack;

unsigned int flags;

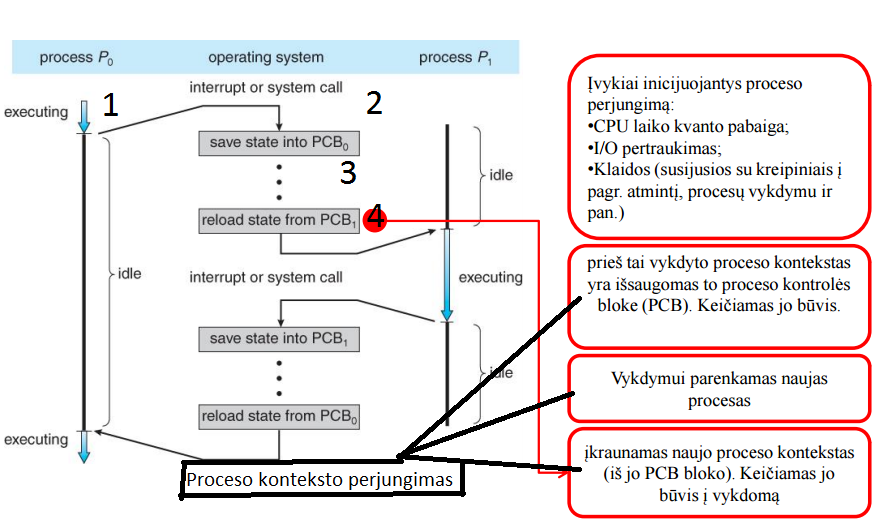
int prio, static\_prio;

struct list\_head tasks;

struct mm\_struct \*mm, \*active\_mm;

4.6 Procesu dispečerės ir planuoklės, paskirtis ir gyd. Metodai

??

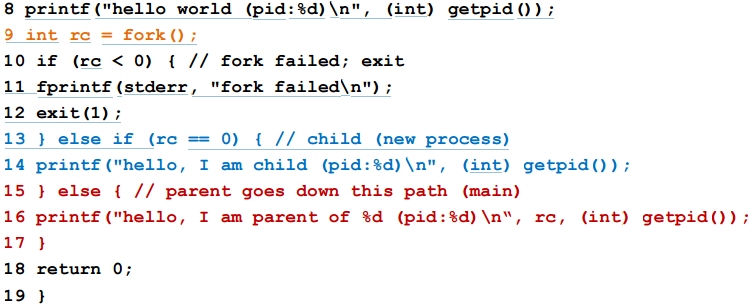
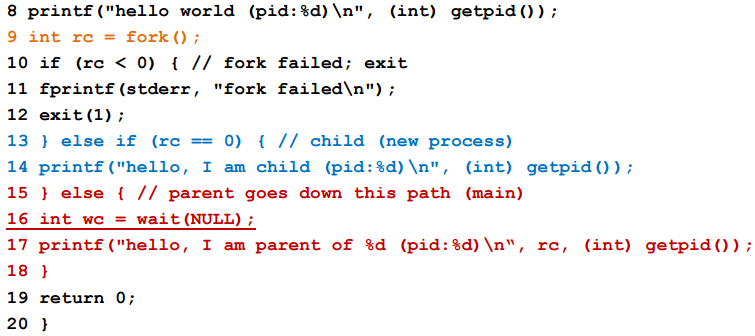
 4.7 Proceso konteksto perjungimas

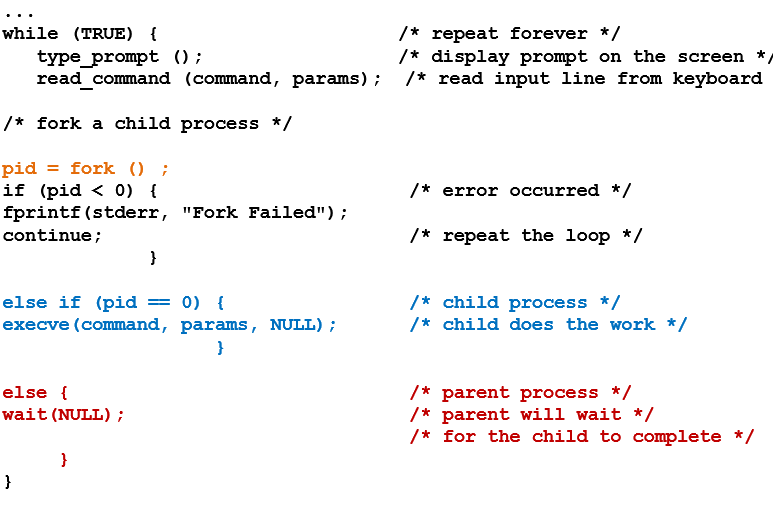
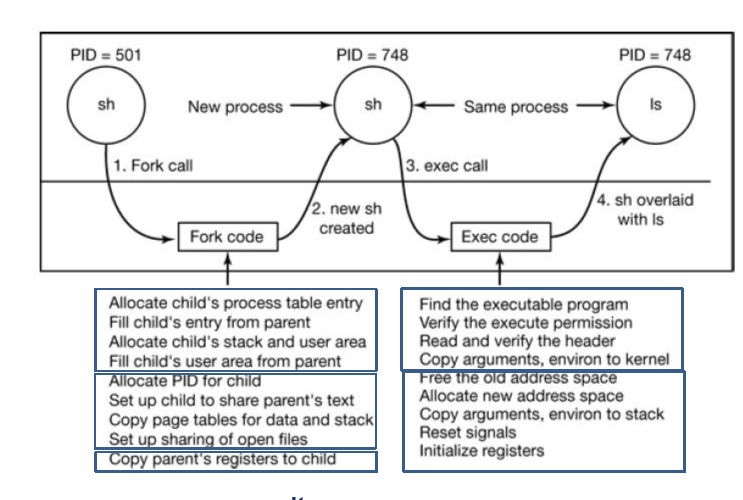
4.8 Proceso vygdymo režimai

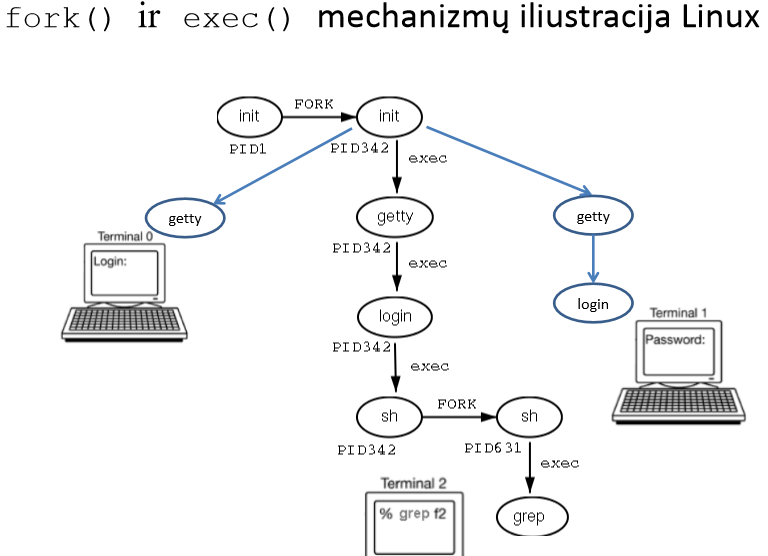
1. Privilegijuotas sistemos rėžimas, susijusia su įrašais sisteminėse duomenų lentelėse ir t.t.

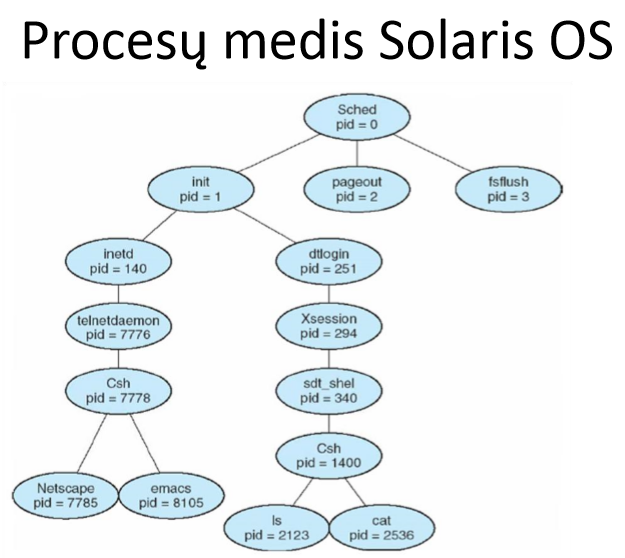
2. Vartotojo rėžimas, neprivilegijuotas darbo rėžimas.

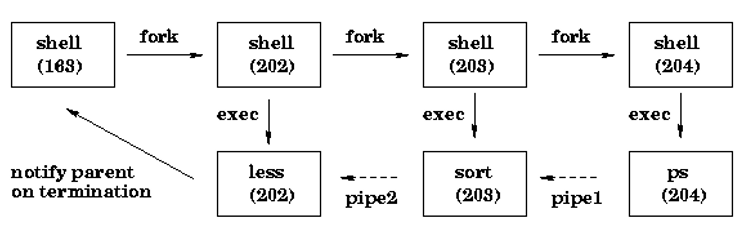
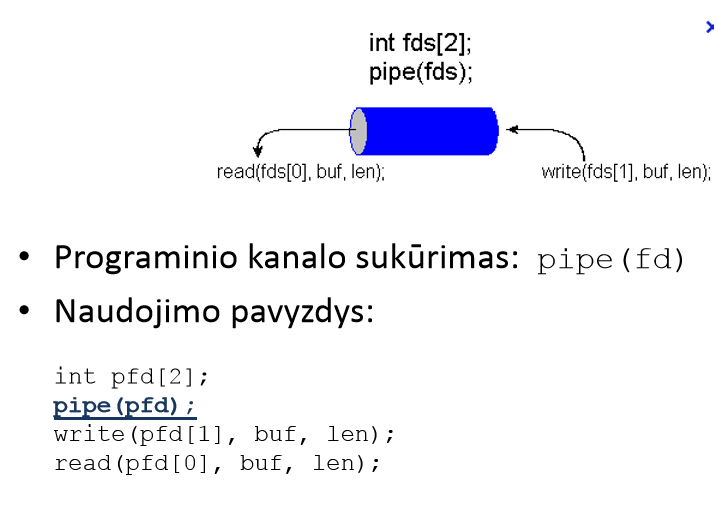
Vygdomo rėžimo pasikeitimas: Pasikeičia tik tada kai reikia atlikti privilegijuotus veiksmus.

4.9 Procesų API









* UNIX signalai:

1. Signalo apdorojimas realizuojamas naudojant kreipinį signal():

void (\*signal(int sig, void (\*func)(int)))(int);

• sig - Signalo ID (vardas)

• func - Rodyklė į f-ją, kuri iškviečiama signalo pasirodymo metu. Šioje f-joje aprašomi veiksmai susiję su nurodyto signalo apdorojimu.

2. Signalai gali būti persiunčiami kitam procesui naudojant sisteminį kreipinį kill():

int kill(pid\_t pid, int sig);

• pid - Proceso ID

• sig - Signalo ID (vardas)

5.1 Procesu tipai

Nepriklausomi procesai (Independent process)

– Tai nuosekli programa vykdymo eigoje

– Privatus kontekstas

– Rezultatai priklauso tik nuo įėjimo duomenų

• Konkuruojantys procesai (Concurrent processes)

– Dalinimasis resursais (konkurencija)

– Vykdymas reikalauja planavimo

• Kooperuojantys procesai (Cooperating processes)

– Lygiagretaus principo programa jos vykdymo metu

– Bendrai naudojamas kontekstas ar jo dalis

– Rezultatai priklauso nuo vykdymo eigos

5.2 Gijos Tipai

Vartotojo gijos ir Branduolio gijos, hibridinės gijos.

Gijų Projektavimo poreikis:

• Gijos daug natūraliau atitinka lygiagrečius skaičiavimus:

naudodami procesą su daugeliu gijų galime gauti puikų serverį.

• Joms nereikia priemonių, leidžiančių komunikacijas tarp procesų,

jos gali komunikuoti per bendrą atmintinę.

• Gijos gali lengvai pasinaudoti nauda, kurią teikia

daugiaprocesorinės sistemos.

5.3 Gijos ir proceso skirtumai/pranašumai.

Panašumai:

– Kaip ir procesai, gijos gali būti sukuriamos, atidedamos ar užbaigiamos.

– Kaip ir procesai, gijos dalinasi procesoriumi (CPU) ir tik viena gija (esant

vienam CPU) yra aktyvi (vykdoma) bet kuriuo laiko momentu.

– Kaip ir procesai, proceso gijos, esant vienam CPU, yra vykdomos nuosekliai.

– Kaip ir procesai, gijos gali sukurti „vaikus“.

– Kaip ir procesų atveju, jei viena branduolio gija yra blokuojama, gali būti

vykdoma kita gija.

• Skirtumai

– Skirtingai nei procesai proceso gijos nėra tarpusavyje nepriklausomos.

– Proceso gijos turi prieigą prie bet kurio adreso proceso erdvėje.

– Proceso gijos yra projektuojamos siekiant, kad jos padėtų viena kitai, o

projektuojant procesus to nėra siekiama, nes jie gali būti skirtingų vartotojų.

5.4 Gijų relizacijos medeliai

Egzistuoja trys galimos gijų realizacijos. Tai:

– Branduolio lygio gijos

• Naudingos esant lygiagretaus darbo galimybei (daug procesorių,

branduolių)

• pav.: Windows 9x/NT/2000, Linux, Solaris,Tru64 UNIX, MacOS X

– Vartotojo lygio gijos

• Planuojamos vartotojo procese

• Branduoliui nematomos

• pav.: POSIX Pthreads, Win32 threads, Java threads

– Kombinacija tarp vartotojo lygio ir branduolio lygio gijų

5.5 Gijų tarpusavio persijungimai

* Proceso registai turiniai išsaugomi vykdyto proceso aplinkoje.
* Registrų turiniai yra įkeliami išnaujo proceso aplinkos.

5.6 Gijų naudojimo teikiami pramašumai

* Gijos leidžia veiksmus atlikti lygiagrečiai.
* Gijos reikia mažiau operacinės sistemos sukurti ir užbaigti.

5.7 Gijos ir signalai

* Sinchroniai (gali atsirasti kaip tiesioginis programos vygdymo rezultatas)
* Asinchroniniai (Nepriklausomi nuo programos,išoriniai proceso atžvilgiu)

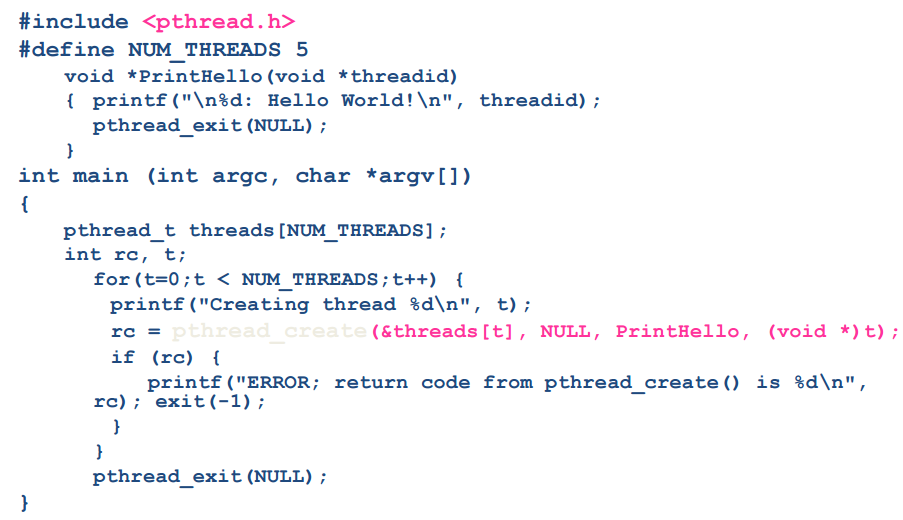
5.8 Gijų API

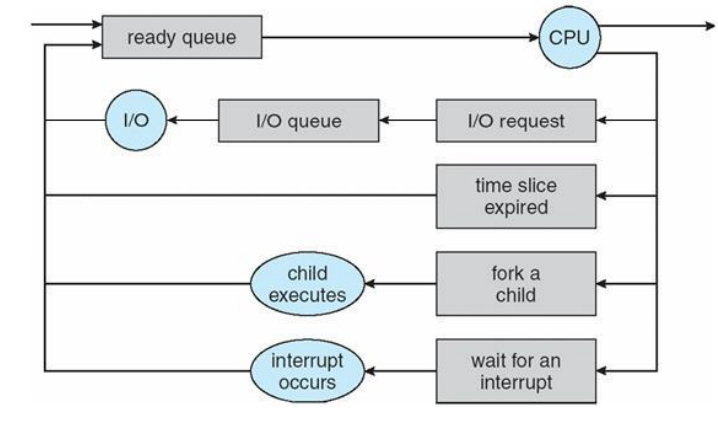
* Pthread\_create – Gijos sukūrimas

– Sukuria naują giją ir padaro ją vykdomąja, visos gijos yra lygiavertės.

– gijos ID grįžta per thread argumenta.

* Pthread\_join – sujungimas
* Pthread\_exit – Gija baigiasi

– Gija baigia iškviestos funkcijos veiksmus



6.1 Proceso vykdymo planavimo savoka/paskirtis.

6.2 Planavimo tipai

* Ilgalaikis planavimas:

Aktyvuojama:

– Kažkuriam iš procesų pasibaigus.

– Kai procesorius prastovi ilgiau nei tam tikrą nustatytą laiko intervalą.

* Vidutinio laiko planavimas:

Aktyvuojama:

– Procesui perėjus į blokuoto būseną

– Nesant pakankamai vietos visiems procesams pagrindinėje atmintinėje

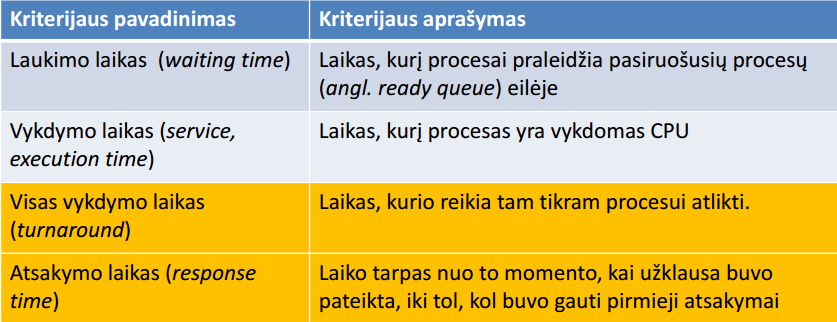
* Trumpalaikis planavimas:
* Sprendžia, kurio proceso vykdymui bus

skiriamas CPU- tai CPU planuoklė.

6.3 Procesų resursų perėmimo būdai

* Prevencinis planavimas: pirmumo planavimas būtų teikiama pirmenybė. Didžiausias prioritetas procesas visada turi būti procesą, kuris šiuo metu yra panaudotos.
* Neprevencinis planavimas: kai procesas patenka į važiavimo būseną, šios proceso valstybė nėra išbrauktas iš tvarkaraščio, kol jis baigia savo tarnavimo laiką.

6.4 Planavimo metrika



6.5 Planavimo Politika-Tikslai

• Minimizuoti \*visą+ atsakymo laiką:

– Minimizuoti proceso apdorojimo laiką sistemoje (laikas pasiruošusių

procesų eilėje, išskirti reikiamą CPU kvanto dydį)

• Maksimizuoti pralaidumą (angl. throughput):

– Minimizuoti papildomas sąnaudas (pavyzdžiui proceso perjungimo

konteksto skaičių)

– Efektyviai naudoti esamus resursus (ne tik CPU, bet ir diskas, atmintis

ir t.t.)

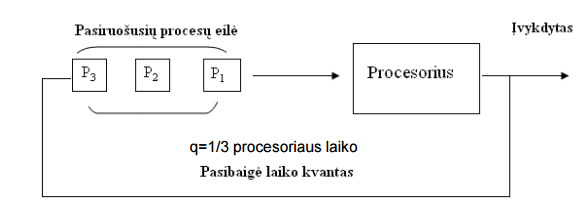
• Vykdyti sąžiningą resursų paskirstymą

– Dalinti CPU resursus sąžiningai – subjektyvu, tačiau bendru atveju

vertinamas bendras vartotojo pasitenkinimas sistemos darbu

– Daugiau sąžiningumo – didesnis atsakymo laikas.

6.6 Planavimo algoritmai



RR ir FCFS (FCFS neturi q)

* SRTF, FCFS ir RR palyginimas

• Tarkime turime tris procesus:

– A ir B – skaičiavimo procesai (CPU bounded, 1 savaitės trukmės)

– C – interaktyvus procesas (I/O bounded, 1ms CPU, 9ms I/O)

• Jei apibrėžtu laiko tarpu turime tik vieną procesą, tai:

– C atveju – 90% naudojamas kietasis diskas.

– A arba B atveju – 100% naudojami CPU resursai.

• FCFS atveju – A arba B gavę CPU, jį naudos savaitę

• RR arba SRTF atveju.

6.7 Sąžiningas planavimas

• Reikalingas kompromisas: sąžiningumas didina atsakymo laiką

• Sąžiningumo mechanizmo realizacija:

– kiekvienai eilei paskirti fiksuotą dalį CPU laiko  ne visuomet

pasiteisina (kas atsitiks, jei turim 100 trumpų procesų ir vieną ilgą?)

– Didinti prioritetą tiems procesams, kurie negauna CPU laiko

(naudojama UNIX tipo OS)  neaišku kaip dažnai ir iki kokio prioriteto

reikia didinti, kad būtų optimalu.