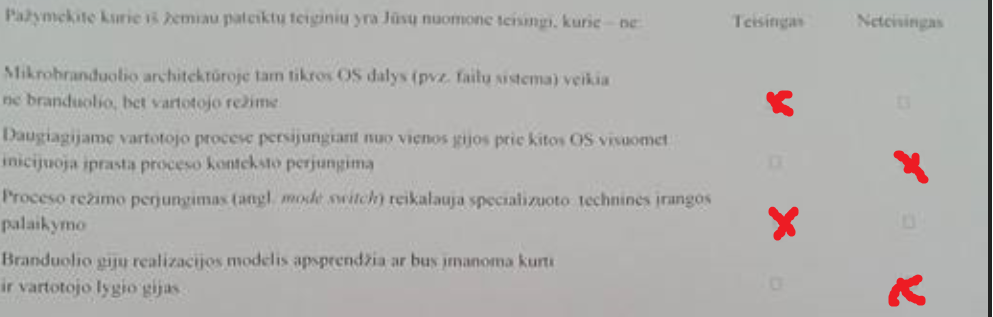
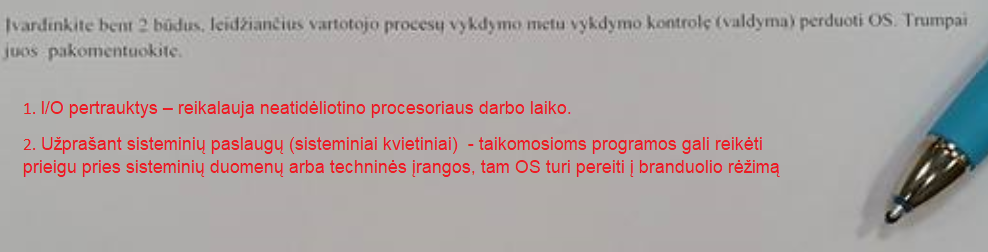
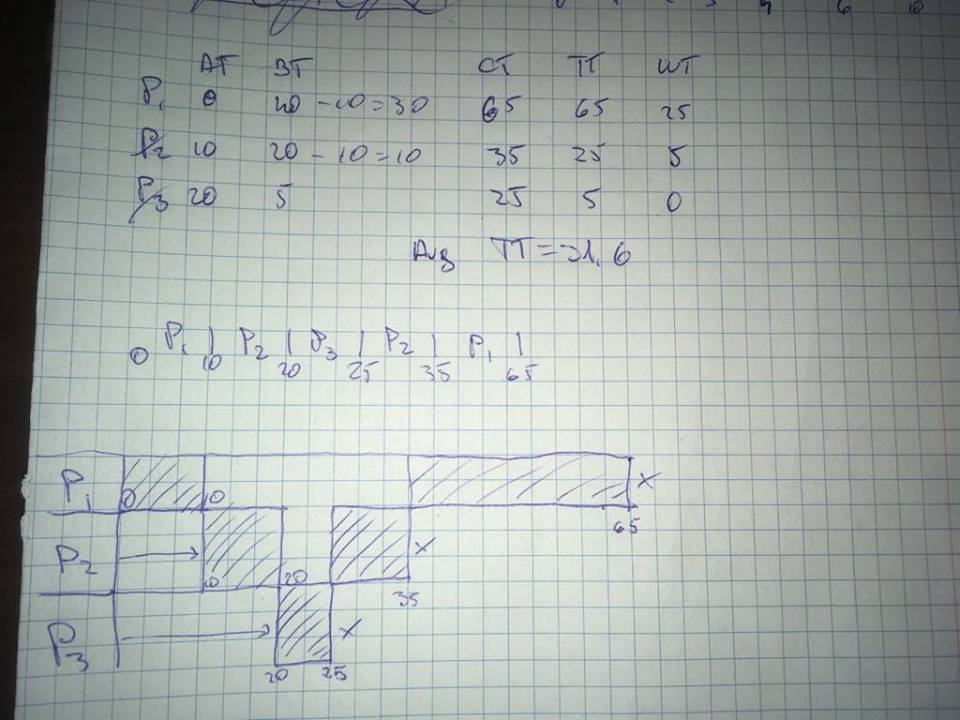
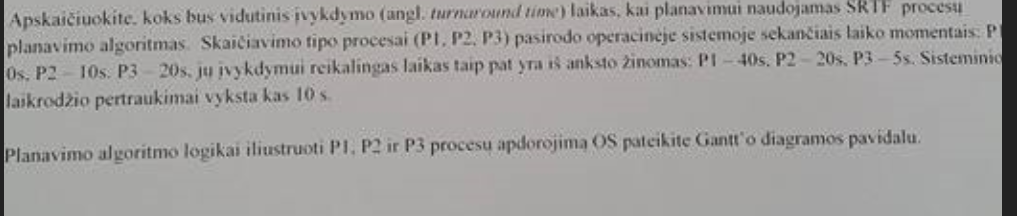
1 var.  
1uzd



2uzd



3 uzd

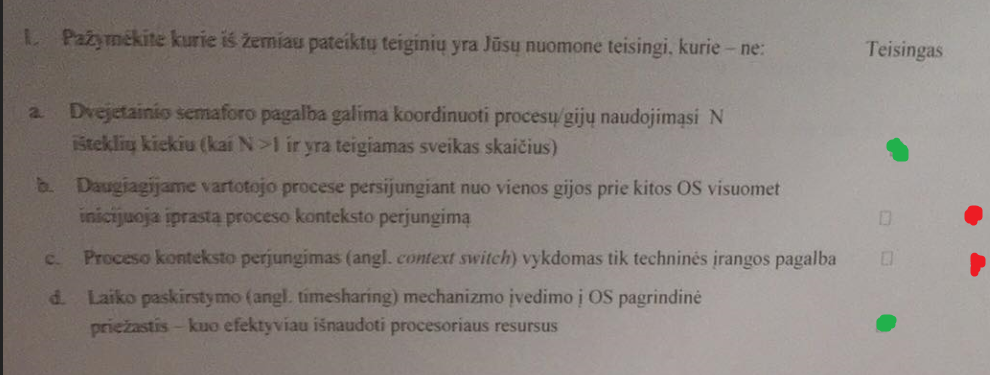


Deja uzduoties fotkiu neturejau tai is atminties ats parase 4-5 uzduotims  
Sinchronizuoti dviejų procesų su skaitmeninių semaforu negalima, tam naudojamas dvejetainis semaforas. Skaitmeninis naudojamas kai procesų yra daugiau (pvz filosofų problema).

Problemos sprendimo būdas nekorektiškas, nes nėra semaforo ar kokios kitos apsaugos kritinei sekcijai.

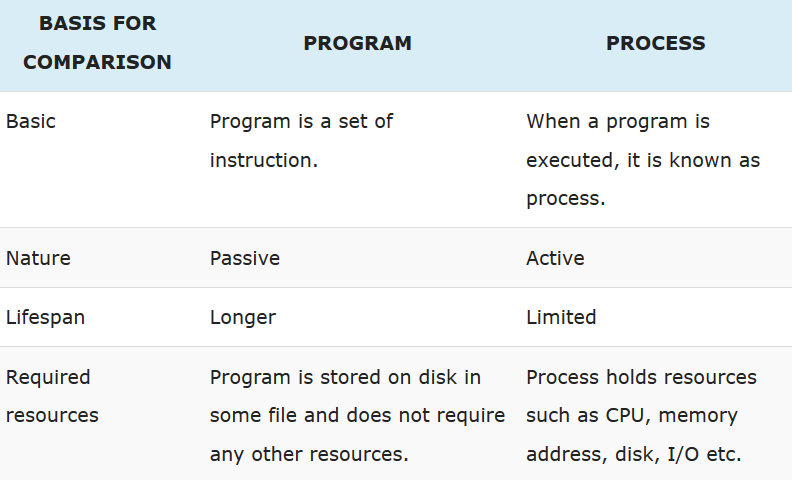
Multiprogramavimui reikalingas procesų dispečeris-planuoklis

Kitas varijantas.  
1. Uzd.

  
raudona tai tipo neteisingas nes sudarankis esu nenufotkinau normaliai lapo.  
  
2uzd.

2. A) Kuo programa skiriasi nuo proceso? B) Kokie žingsniai turi būti atlikti, kad programa taptu procesu?

Programa ir procesas yra susiję terminai. Pagrindinis skirtumas tarp programos ir proceso yra tai, kad programa yra instrukcijų grupė atlikanti nurodytą užduotį, o procesas yra vykdoma programa. Nors procesas yra aktyvus subjektas, programa laikoma pasyvia.



B) 1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas

2. Proceso identifikatoriaus (PID) sukūrimas

3. Vykd. dvejet. programos kodo užkrovimas į atmintį

4. Atminties išskyrimas stekui

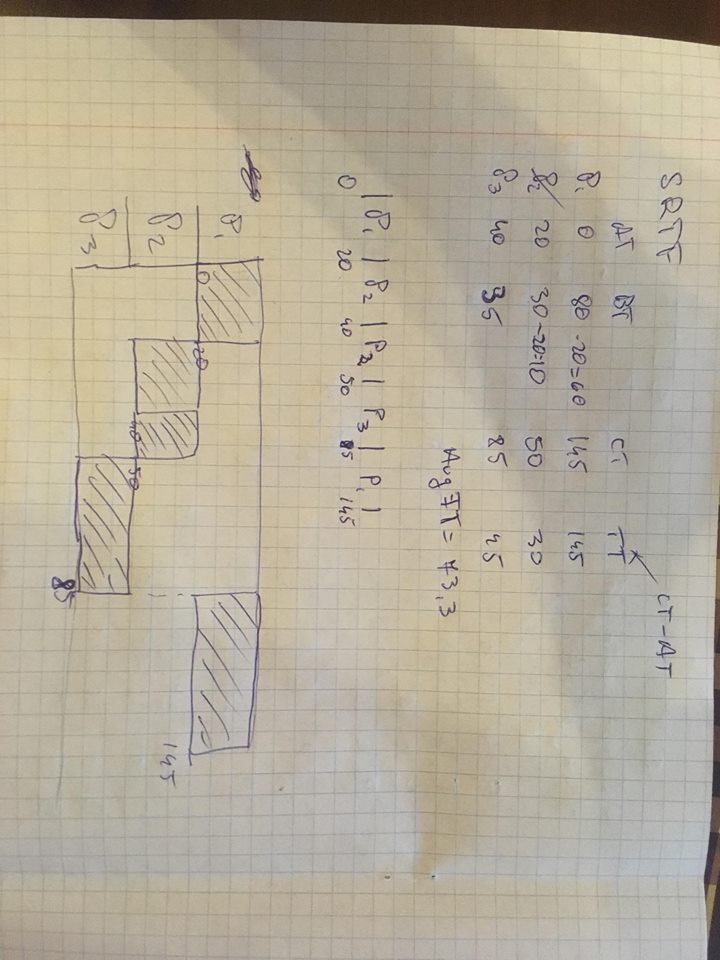
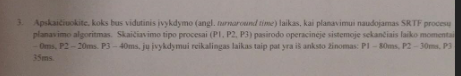
5. Atminties išskyrimas heap sričiai

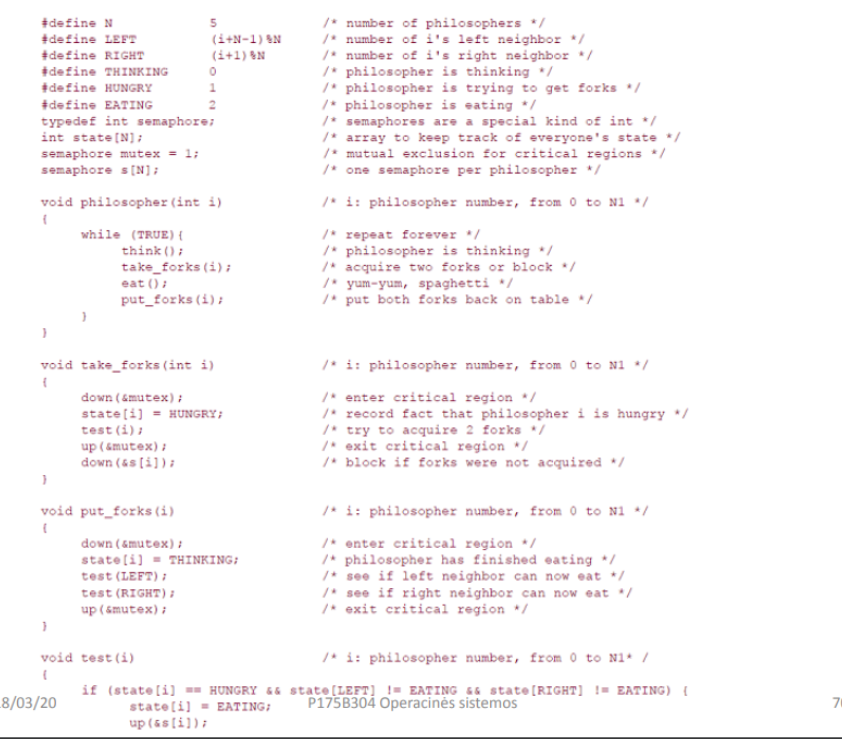
6. Veiksmų susijusių su I/O inicializavimas

7. PCB užpildymas

8. Pirmosios progr. komandos užkrovimas (entry point)

3uzd.

  
  
4. Užduotis

Kas butu jeigu putforks metode state[i] = thinking nukeltumem po abiem test(i). Priedas zemiau. (P.S PRIEDAS:  


Priedus issitampysit kai ziuresites taupau vieta

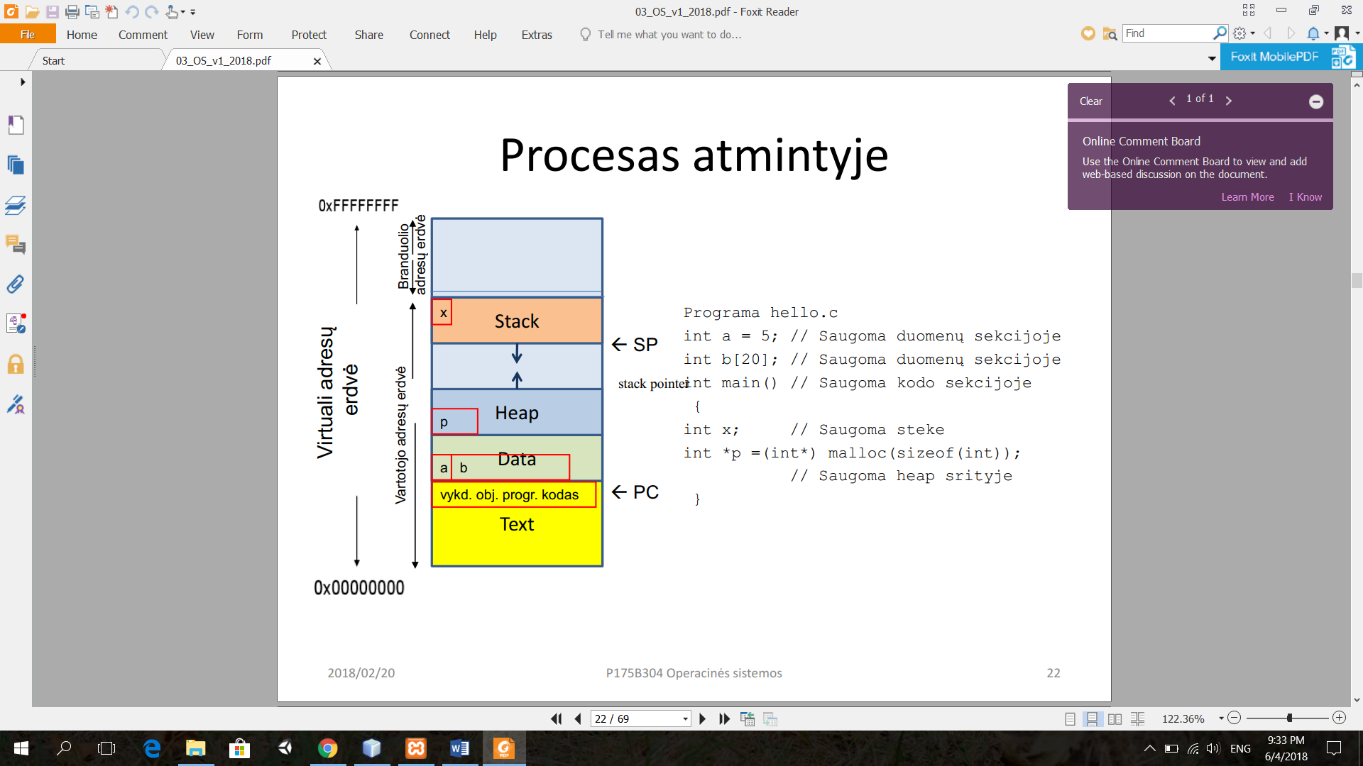
Ats: Mazdaug.

Kai filosofas paima šakutes, jo state pasidaro HUNGRY ir jis žiūri, ar iš kairės ir dešinės filosofas nevalgo, o tik tada padaro state EATING. Po to, filosofo state pasikeičia į THINKING.

Tačiau, kai THINKING state nukeliamas dvejomis eilutėmis žemiau, tada jo state vis dar būna EATING. Todėl kai jis pirmiau pradeda žiūrėt, ar kaimynas iš kairės nevalgo, o pas patį vis dar state EATING, gaunasi užburtas ratas(kitaip badavimas <- cia yra keyword sitai uzduociai) ir filosofo state nepasidarys normaliai THINKING.

5uzd:  
Buvo tiesiog kodas ir reikejo parasyt kas vyksta atminti.

Va taip kaip nuotraukoj.



Kitas varijantas.

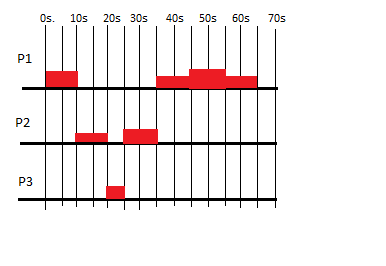
1. Pažymėkite kurie iš žemiau pateiktų teiginių Jūsų nuomone yra teisingi, kurie - ne:
   1. Bet kokia operacinė sistema veikia tik branduolio režime Neteisingas
   2. Vienos gijos operacinėje sistemoje galima realizuoti daugiagijį vartotojo procesą Teisingas
   3. Proceso konteksto perjungimą vykto tik techninė įranga Neteisingas
   4. Proceso kontrolės blokas(PCB) sukuriamas sukompiliavus programą Neteisingas
2. Išvardykite tuos procesų planavimo algoritmus, kuriuos naudojant niekuomet neiškils badavimo situacija.

FCFS – pirmas pasirodęs procesas yra aptarnaujamas pirmas (FIFO)

RR – sprendžia FCFS konvojaus efekto problemą. Tinka, nes kiekvienas procesas gauna vienodą laiko kvantą.

1. Apskaičiuokite koks buvo vidutinis įvykdymo laikas, kai planavimui naudojamas SRTF procesų planavimo algoritmas. Skaičiavimo tipo procesai (P1, P2, P3) pasirodo operacinėje sistemoje sekančiais laiko momentais: P1 - 0s, P2 - 10s, P3 - 20s, jų įvykdymui reikalingas laikas taip pat yra iš anksto žinomas: P1 - 40s, P2 – 20s, P3 – 5s. Sisteminio laikrodžio pertraukimai vyksta kas 10 s.

Planavimo algoritmo logikai iliustruoti P1, P2 ir P3 procesų apdorojimą OS pateikite Giant‘o diagramos pavidalu.



RTp1 =65s.

RTp2 = 25s.

RTp3 = 5s.

RTavg = (65s +25s +5s)/3 = 95s/3 = 31,66s

Vidutinis įvykdymo laikas: 31,66s.

1. Ar dvejetainio semaforo pagalba galima užtikrinti vienalaikį N procesų ( kai N – bet koks teigiamas skaičius) naudojimąsi tam tikru resursu. Atsakymą pagrįskite.

Taip, galima , nes semaforas yra operacinės sistemos teikiamas įrankis, leidžiantis užtikrinti sinchronizaciją. Semaforas s gali būti traktuojamas kaip sveikojo tipo kintamasis, su kuriuo gali būti atliekamos dvi operacijos: įjungimo(signal(s)) ir išjungimo(wait(s)) . Procesai norėdami patekti į kritinę sekciją turi patikrinti semaforo reikšmę. Jeigu yra prasidėjusi operacija signal() arba wait(), tai joks kitas procesas negali prieiti prie to semaforo šios operacijos metu.

Semaforas:

* Užtikrina išskirtinumą - kad kelios gijos vienu metu nenueitų į kritine sekciją
* Užtikrina sinchronizaciją – kad gijos atliktu veiksmus tam tikra tvarka.

1. Pridėto priedo 2-ame lape jums pateiktas vartotojo – gamintojo problemą sprendžiantis programos fragmentas. Išanalizuokite jį ir atsakykite į sekančius klausimus:
   1. Ar tai yra korektiškas šios problemos sprendimo būdas?

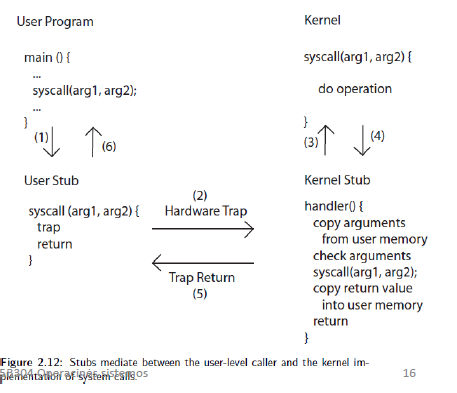
Korektiškas/Nekorektiškas

* 1. Jei taip – tai kodėl, jei ne – tai kokios problemos gali kilti sprendžiant šį uždavinį tokiu būdu?

Korektiškas, nes nėra situacijos kai užmiega abu.

1. Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant bash shell‘o komandą cd (jos vykdymo metu). **Paaiškinkite kodėl.**

Gali, nes yra iškviečiama pertrauktis.



Kitas varijantas.

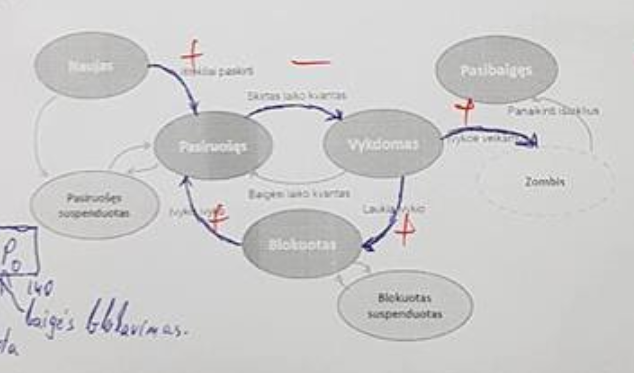
1. Užduotis
   1. **Mikrobranduolio architektūroje tam tikros OS dalys veikia ne branduolio, bet vartotojo režime.** Tiesa, nes OS dalys kaip failų sistema, cpu scheduller, memory management turi savo adresų erdvę. Vartotojo sukurtas procesas kreipiasi į tam tikras OS dalis.
   2. **Daugiagijame vartotojo procese, persijungiant nuo vienos gijos prie kitos, OS visuomet inicijuoja įprastą proceso konteksto perjungimą.** Netiesa, nes proceso gijos dalinasi proceso adresų erdve (visu proceso PCB), todėl nėra būtina kitos gijos veiklos suspenduoti.
   3. **Proceso režimo perjungimas (mode switch) reikalauja specializuoto techninės įrangos palaikymo.** Tiesa, nes vartojo režimas yra perjungiamas į kernel režimą, o tam reikia CPU?
   4. **Branduolio gijų realizacijos modelis apsprendžia, ar bus įmanoma kurti ir vartotojo lygio gijas.** Netiesa, nes kiekvienam vartojo lygio gijai yra priskiriama bent viena branduolio lygio gija.
2. Užduotis

**Įvardinkite bent 2 būdus leidžiančius vartotojo procesų vykdymo metu vykdymo kontrolę (valdymą) perduoti OS. Trumpai juos pakomentuokite.**

* 1. Programos vykdymo metu kviečiant system calls (kaip open(), read(), write()...) perjungia user mode į kernel mode.
  2. Programos vykdymo metu atsiradusių klaidų (exceptions), tokių kaip dalyba iš nulio ar pan. Perjungia user mode į kernel mode exception handlerį.

1. Užduotis

**Pažymėkite tuos proceso būsenų grafo lankus, kuriuose aktyvuojamas CPU planuotojas procesus vykdant pagal procesų planavimo algoritmą, kai vykdymui parenkamas procesas su mažiausiu likusiu aptarnavimo laiku (SRTF).**



Proceso perjungimo metu, CPU planuotojas paskiria išteklius sustabdytam procesui ir visuomet jį **paruošia** (negali iš karto vykdyti). Vykdomas procesas gali būti **blokuotas**, tuomet perjungia vėl į kita procesą. Procesui pasibaigus vykdyti, jis tampa **zombiu** ir CPU parenka sekantį procesą vykdymui.

1. Užduotis

**Ar skaitmeninio semaforo pagalba galima užtikrinti procesų tarpusavio atskyrimą? Atsakymą pagrįskite.**

Šito nežinau 🙁

MANAU, kad ne, nes skaitmeninis semaforas yra naudojamas daugelių procesų resursų paskirstymui. T.y. kiekvienam procesui suteikiamas reikiamas kiekis resursų, jeigu jų yra laisvų. Tarkime, visas procesų skaičius - 30, o resursų yra tik 10-čiai, tai 20 procesų laukia eilėje. Pasibaigus vienam procesui, priskiriami resursai likusiems. Jeigu tai būtų dvejetainis semaforas, tada taip, nes vienas procesas būtų užlokintas, kol kitas pasibaigtų.

1. Priedas
2. Užduotis

**Kokie mechanizmai reikalingi norint operacinėje sistemoje įgyvendinti multiprogramavimo koncepciją? Kodėl?**

* 1. Procesų planavimo mechanizmas – siekiant maksimaliai išnaudoti CPU laiką. Pavyzdžiui, kol atliekama I/O užduotis, kuriam CPU išteklių nereikia, inicijuojamas konteksto perjungimas ir atliekamas sekantis procesas. Pagrindinis tikslas - maksimaliai užkrauti CPU, kol yra pasiruoštų procesų vykdymui.
  2. Atminties valdymas – labai svarbu tinkamai išnaudoti atmintį. Vykdant procesus, gali atsirasti atminties plyšių (per mažų vietų fizinėje atmintyje kitiems procesams). Tai padeda išspręsti fragmentacija arba virtualiosios atminties valdymas.
  3. Minimalus informavimo apie I/O pabaigą mechanizmas.