Koliokviumo užduočių-klausimų pavyzdžiai

Konsultacija

2018-03-19
2018-03-20

Koliokviumo data, formatas

- Balandžio 9-10 d. (pirmadienis-antradienis)
- Trukmė 45 min.
- Per paskaitą – du srautai
- Būtina išankstinė registracija Moodle

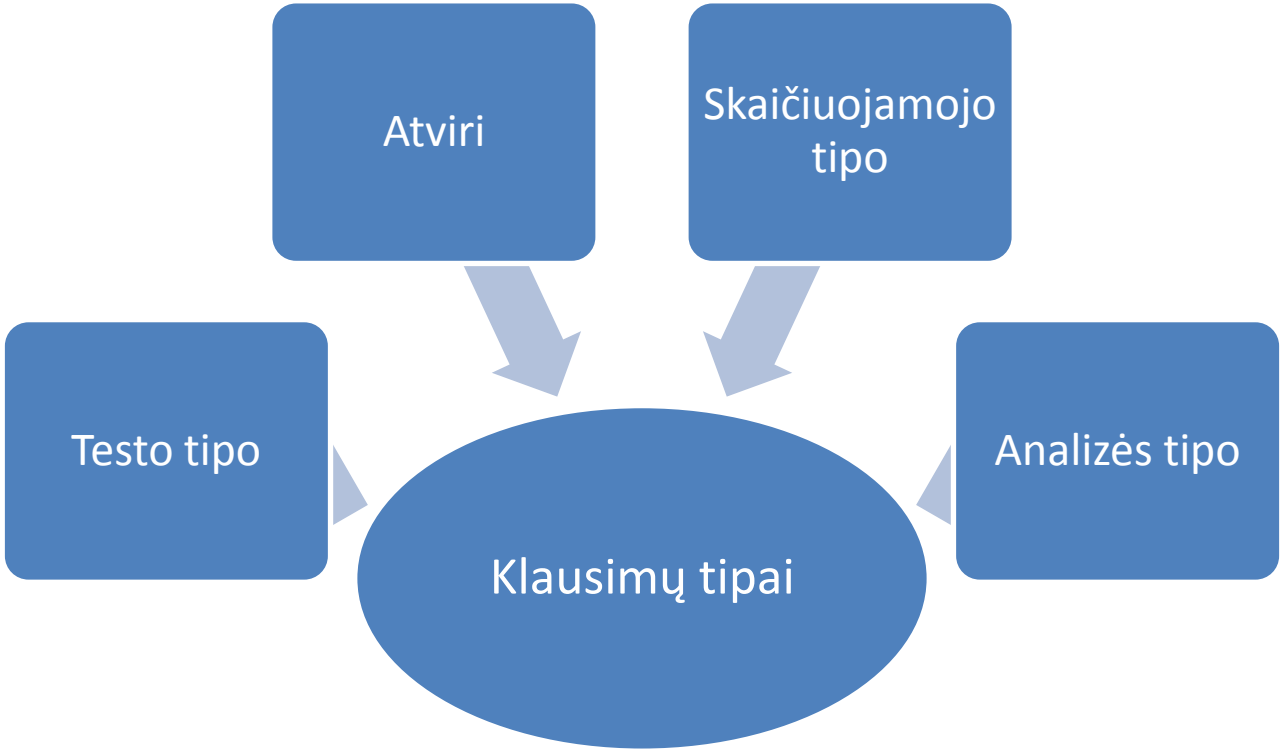
- Pakartotinis atsiskaitymas - per paskutinę paskaitą.

Temos

OS teorinės medžiagos temos koliokviumui

Eil. Nr.	Tema	Smulkesni su tema susiję klausimai
	ĮVADAS Į OS (01T – 02T)	
1.	Įžanga	• OS koncepcija • Kompiuterio techninė įranga (procesorius, atmintinė, atmintinės įrenginių hierarchija, įvedimo/išvedimo įrenginiai, magistralė) • Bazinis programos vykdymo ciklas • Pertraukys (programinės ir aparatinės pertrauktys, jų apdorojimo logika)
2.	OS apžvalga	• OS paskirtis ir funkcijos • Daugelio užduočių vykdymas (angl. *multitasking*) • OS tobulinimo poreikiai • OS vystymosi istorija. • OS savybės skirtingais jų vystymosi etapais. • OS komponentai (procesų valdymas, atmintinės valdymas, I/O sistemos valdymas, failų valdymas, apsaugos sistema, darbas tinkle) ir paslaugos. • Sisteminiai kvietiniai.
3.	OS architektūros	• OS architektūros sąvoka. • Skirtingų OS architektūrų veikimo principai, jų privalumai ir trūkumai. • Skirtingų architektūrų OS pavyzdžiai
	PROCESŲ VALDYMAS OPERACINĖJE SISTEMOJE (03T – 07T)	
4.	Procesai	• Proceso sąvoka/apibrėžimas. Proceso komponentai. • Pagrindinės procesų valdymą apimančios funkcijos. • Procesas atmintyje. • Proceso būsenų diagramos (3, 5, 7 būsenų). Zombio būsena. Tipinė UNIX tipo OS procesų būsenų diagrama. • Proceso valdymui naudojamos duomenų struktūros (proceso kontrolės blokas, procesų lentelė, procesų eilės). • Procesų dispečerės-planuoklės paskirtis, jos komandų vykdymo periodai (t.y. kada ji aktyvuojama). • Proceso konteksto perjungimas (angl. *context switch*). Detali perėjimo nuo vieno proceso prie kito procedūra. • Procesų vykdymo režimai, vykdymo režimo pasikeitimas (angl. *mode switch*) • Procesų API (fork(), wait(), [illegible] paskirtis, jų taikymo pavyzdžiai, procesų medis)

Klausimų/užduočių tipai



Testo tipo klausimai (1)

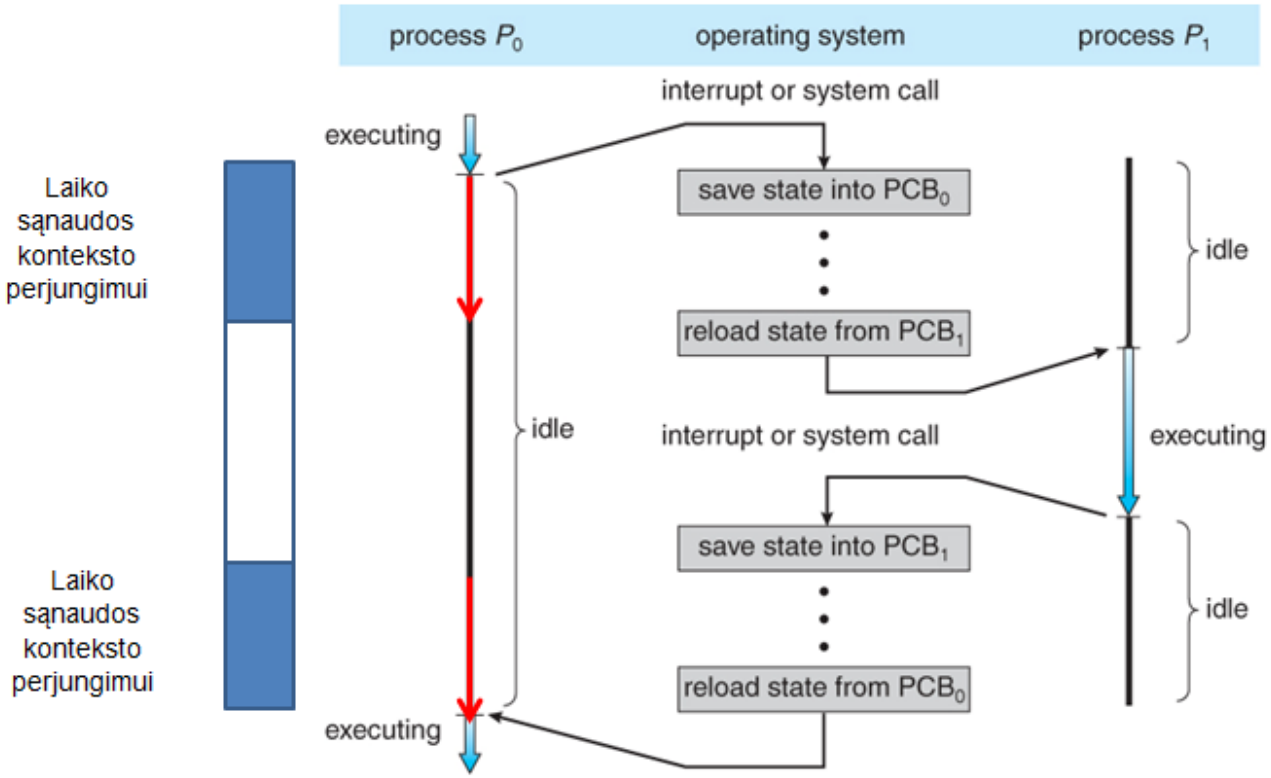
Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
***Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui**	☐	☐
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	☐
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	☐
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

Testo tipo klausimai (1.1)

Konteksto perjungimo skaičius turi/neturi įtakos sistemos efektyvumui

Perėjimo nuo vieno proceso prie kito procedūra



- Proceso konteksto saugojimo metu neišnaudojami CPU resursai;
- Laiko, reikalingo konteksto perjungimui, sumažinimui siūlomi ir įvairūs techniniai sprendimai;

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	☐
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	☐
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

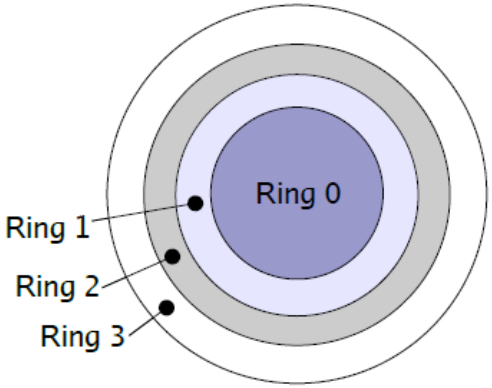
Testo tipo klausimai (1)

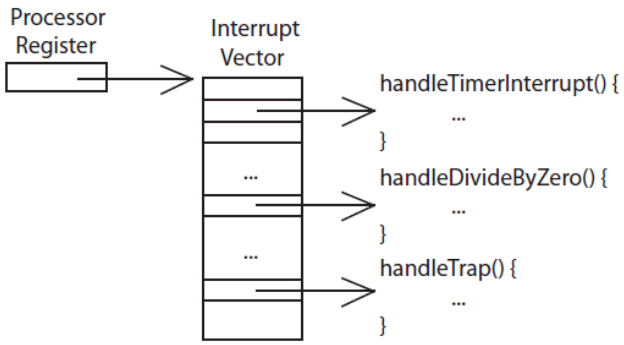
Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

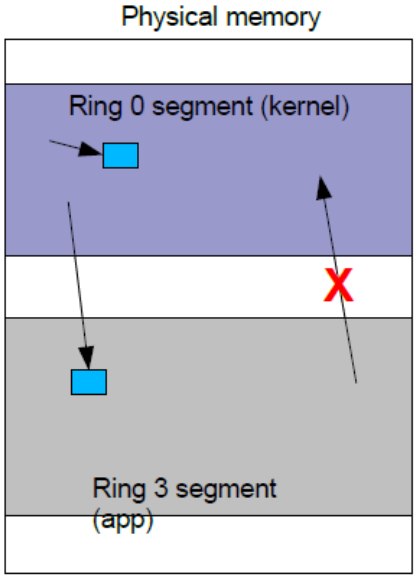
	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
***OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų**	☐	☐
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	☐
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

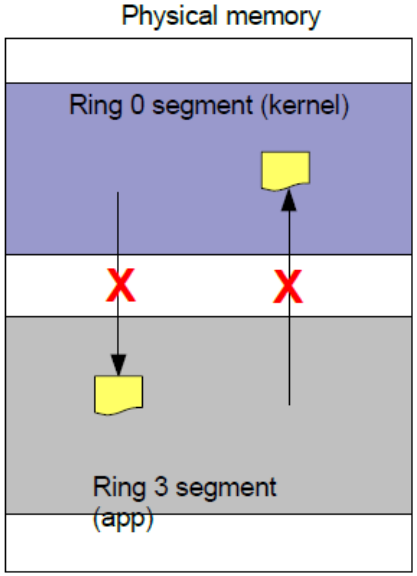
Testo tipo klausimai (1.2)

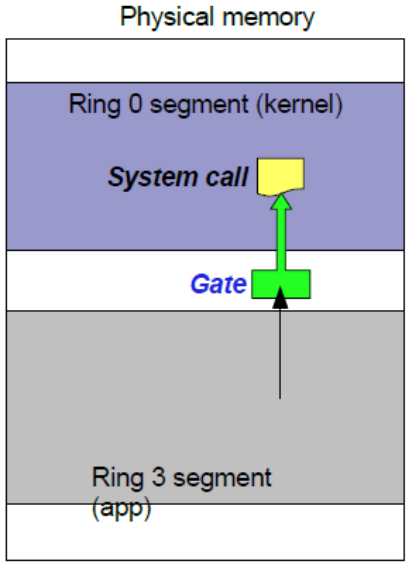
OS instrukcijos vykdomos/nevykdomos tokiu pačiu režimu kaip ir vartotojo procesų











Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	☐
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

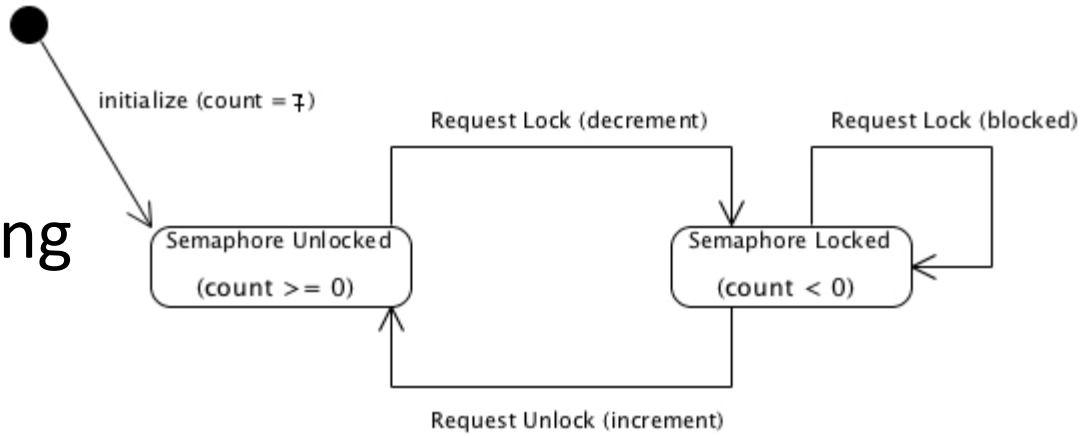
	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
***Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs**	☐	☐
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

Testo tipo klausimai (1.3)

Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs/netapatūs

- Aktyvaus laukimo užraktas (spinlocks)
- Dvejetainis semaforas (Mutexes, binary semaphore)

- Skaitmeninis semaforas (counting semaphore)




```
int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
```

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	✓
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	✓
***Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)**	☐	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

Testo tipo klausimai (1.4)

Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant bash shell'o komandą cd (jos vykdymo metu)

- truss rezultatai (`$truss –p pid; kitam terminale $cd /`)

```
sigaction(SIGINT, 0xFEFFE320, 0xFEFFE380)       = 0
sigaction(SIGTERM, 0xFEFFE320, 0xFEFFE380)      = 0
sigaction(SIGQUIT, 0xFEFFE320, 0xFEFFE380)      = 0
sigaction(SIGALRM, 0xFEFFE320, 0xFEFFE380)      = 0
sigaction(SIGTSTP, 0xFEFFE320, 0xFEFFE380)      = 0
sigaction(SIGTTOU, 0xFEFFE320, 0xFEFFE380)      = 0
sigaction(SIGTTIN, 0xFEFFE320, 0xFEFFE380)      = 0
sigaction(SIGWINCH, 0xFEFFE320, 0xFEFFE380)     = 0
sigaction(SIGINT, 0xFEFFE310, 0xFEFFE390)       = 0
time()                                          = 1427204067
chdir("/")                                      = 0
sigaction(SIGINT, 0xFEFFEC70, 0xFEFFECF0)       = 0
time()                                          = 1427204067
lwp_sigmask(SIG_SETMASK, 0x06820000, 0x00000000, 0x00000000, 0x00000000) = 0xFFBFFEFF [0xFFFFFFFF]
ioctl(255, TIOCGSID, 0xFEFFE34C)                = 0
getsid(0)                                       = 3485
ioctl(255, TIOCSPGRP, 0xFEFFE384)               = 0
lwp_sigmask(SIG_SETMASK, 0x00000000, 0x00000000, 0x00000000, 0x00000000) = 0xFFBFFEFF [0xFFFFFFFF]
sigaction(SIGINT, 0xFEFFE310, 0xFEFFE390)       = 0
lwp_sigmask(SIG_SETMASK, 0x00000002, 0x00000000, 0x00000000, 0x00000000) = 0xFFBFFEFF [0xFFFFFFFF]
ioctl(0, TIOCGWINSZ, 0xFEFFE2B8)                = 0
```

Testo tipo klausimai (1.4)

Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant bash shell'o komandą cd (jos vykdymo metu)

- truss rezultatai (`$truss –p pid; kitam terminale $cd /`)

```
sigaction(SIGINT, 0xFEFFE320, 0xFEFFE380)       = 0
sigaction(SIGTERM, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGQUIT, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGALRM, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGTSTP, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGTTOU, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGTTIN, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGWINCH, 0xFEFFE320, 0xFEFFE380)
sigaction(SIGINT, 0xFEFFE310, 0xFEFFE390)
time()
chdir("/")
sigaction(SIGINT, 0xFEFFEC70, 0xFEFFECF0)
time()
lwp_sigmask(SIG_SETMASK, 0x06820000, 0x0000
ioctl(255, TIOCGSID, 0xFEFFE34C)
getsid(0)
ioctl(255, TIOCSPGRP, 0xFEFFE384)
lwp_sigmask(SIG_SETMASK, 0x00000000, 0x0000
sigaction(SIGINT, 0xFEFFE310, 0xFEFFE390)
lwp_sigmask(SIG_SETMASK, 0x00000002, 0x0000
ioctl(0, TIOCGWINSZ, 0xFEFFE2B8)
```

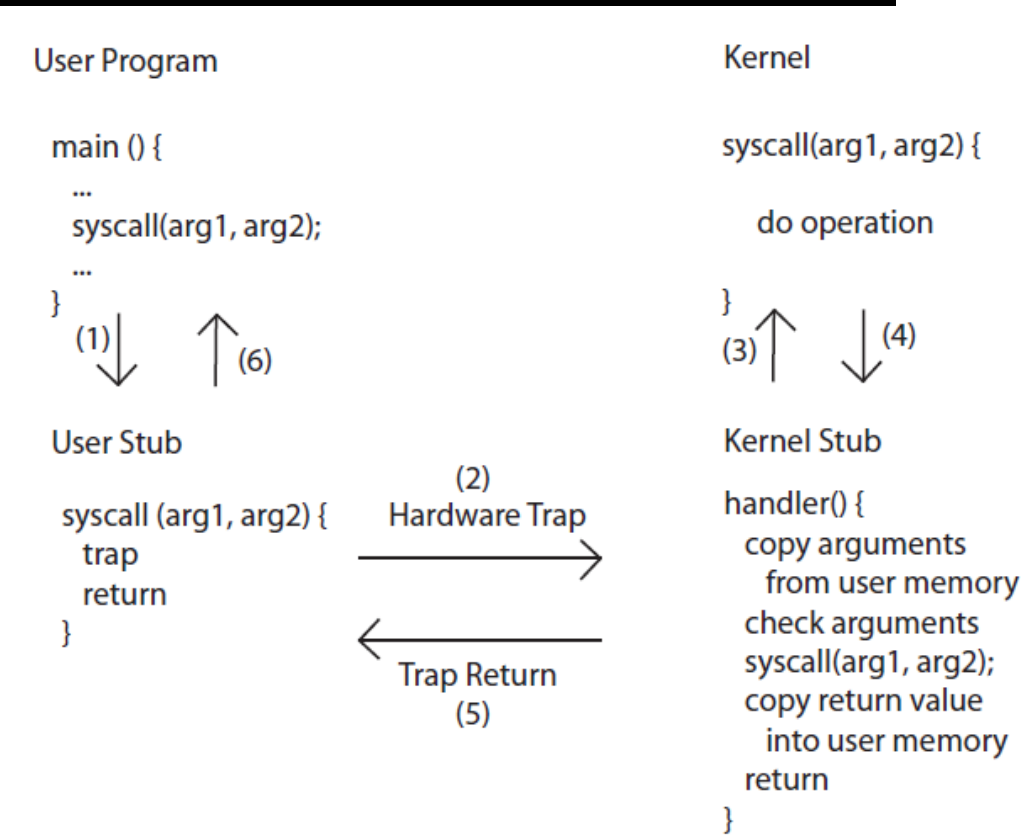


Figure 2.12: Stubs mediate between the user-level caller and the kernel implementation of system calls.

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	✓
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	✓	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	☐

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	☐	✓
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	✓	☐
***SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą**	☐	☐

Testo tipo klausimai (1.5)

Ar SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą

- Planavimo algoritmų įvairovė: FIFO, RR, SPF (SJF), SRT (SRTF, STCF), MLFQ...
- Planavimo politika: atsakymo laikymas, pralaidumas, sąžiningas resursų paskirstymas
- Metrikos: vidutinis įvykdymo laikas (turnaround time), vidutinis atsakymo laikas (response time), pralaidumas (throughput)
- Sąžiningumas: bendru atveju vertinamas bendras vartotojo pasitenkinimas sistemos darbu, daugiau sąžiningumo – didesnis atsakymo laikas
- SRT atveju – galima situacija, kai ilgi procesai gali būti niekad nevykdomi – t.y. ilgesni procesai gali “badauti” – šiuo atžvilgiu sąžiningumo kriterijus neišpildomas.

Testo tipo klausimai (1)

Pažymėkite kurie iš žemiau pateiktų teiginių yra Jūsų nuomone teisingi, kurie – ne:

	T	N
*Konteksto perjungimo skaičius neturi įtakos sistemos efektyvumui	☐	✓
*OS instrukcijos vykdomos tokiu pačiu režimu kaip ir vartotojo procesų	☐	✓
*Užraktai ir skaitmeniniai semaforai savo veikimo principu yra tapatūs	✓	☐
*Ar proceso konteksto perjungimas gali įvykti sistemos vartotojui naudojant *bash* shell'o komandą cd (jos vykdymo metu)	✓	☐
*SRT (SRTF, STCF) planavimo algoritmas užtikrina sąžiningą CPU resursų dalinimą	☐	✓

Testo tipo klausimai (2)

Kuriose būsenose procesas visuomet bus operatyviojoje atmintyje (nevertinant kešo)?

- Būsenoje "Vykdomas" (angl. Running)
- Būsenoje "Pasiruošęs" (angl. Ready)
- Būsenoje "Naujas" (angl. New)
- Būsenoje "Blokuotas" (angl. Waiting)
- Būsenoje "Pasibaigęs" (angl. Terminated)

Testo tipo klausimai (2)

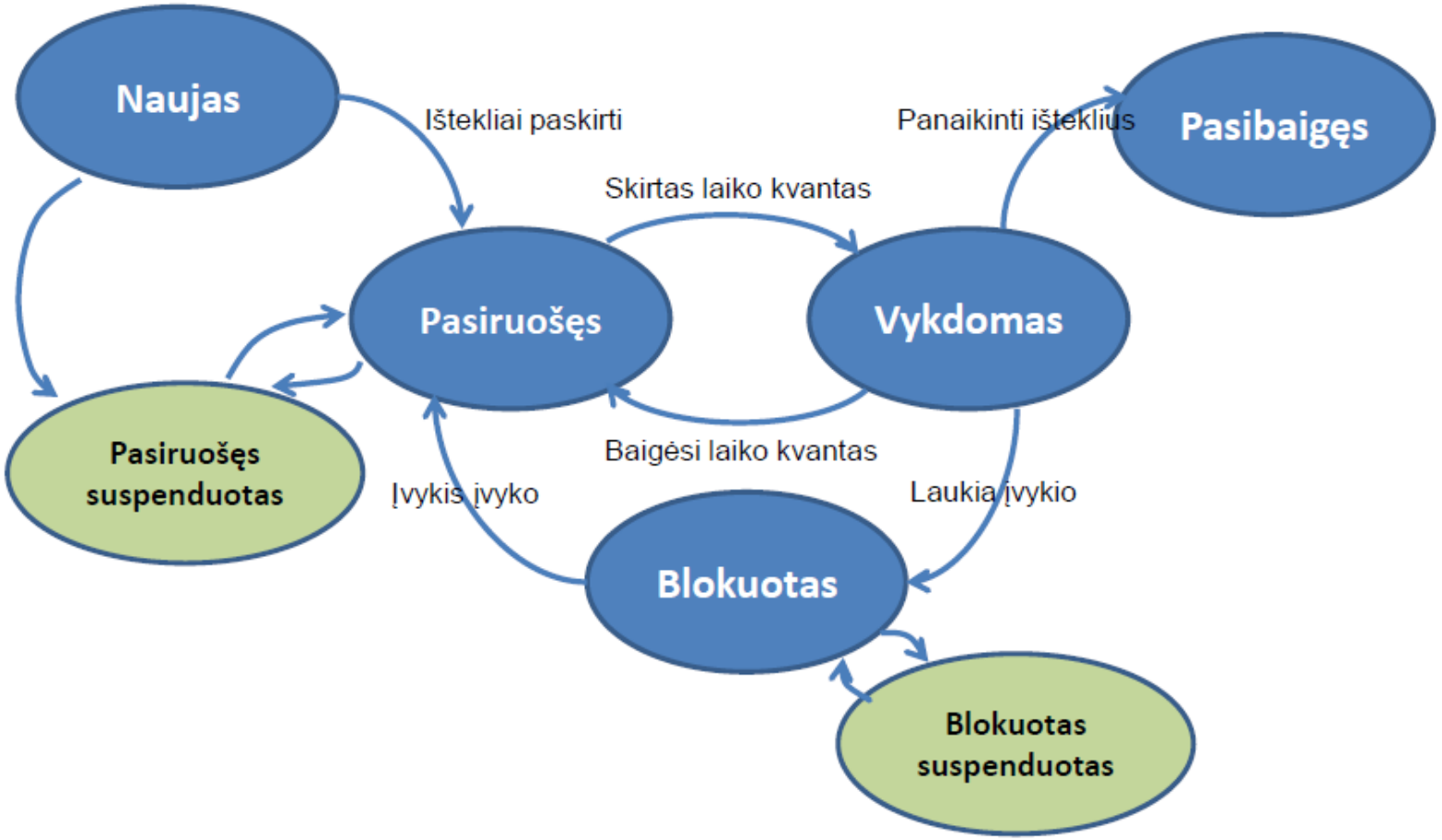
Kuriose būsenose procesas visuomet bus operatyviojoje atmintyje (nevertinant kešo)?

- ✓ Būsenoje "Vykdomas" (angl. Running)
- ✓ Būsenoje "Pasiruošęs" (angl. Ready)
- ○ Būsenoje "Naujas" (angl. New)
- ○ Būsenoje "Blokuotas" (angl. Waiting)
- ○ Būsenoje "Pasibaigęs" (angl. Terminated)

Testo tipo klausimai (2.1, 2.2)

Proceso būsenos

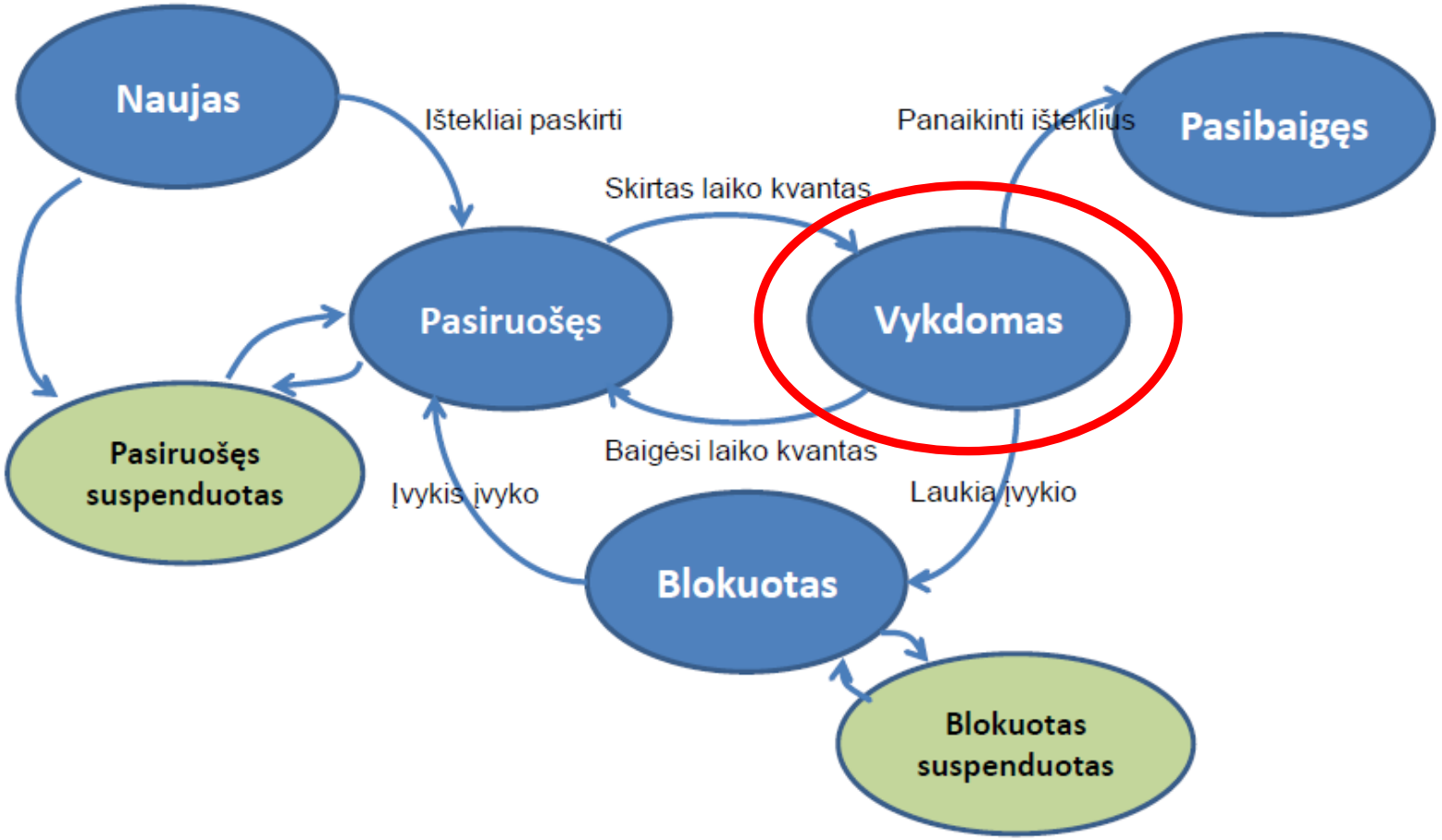
Kokią būseną CPU planuotojas (angl. scheduler) nustato procesui, kurį parenka vykdymui?



Testo tipo klausimai (2.1, 2.2)

Proceso būsenos

Kokią būseną CPU planuotojas (angl. scheduler) nustato procesui, kurį parenka vykdymui?



Pažymėti teisingus teiginius

a. Sukūrus naują vartotojo giją, jai išskiriama ir nauja atmintinės sritis.

b. Daugiagijame vartotojo procese persijungiant nuo vienos gijos prie kitos OS visuomet
inicijuoja įprastą proceso konteksto perjungimą

c. Tiek vartotojo procesui, tiek vartotojo gijai sukurti reikalingi tokie patys sistemos ištekliai.

d. Vartotojo procesą gali sudaryti daug vartotojo gijų

e. Jei vartotojo procesas turi keletą gijų, gali iškilti poreikis šio proceso gijų veiksmus sinchronizuoti

f. Vartotojo gijos kontekstą nusako programos skaitiklis (PC), registrų rinkinys ir dėklo erdvė

Pažymėti teisingus teiginius

a. Sukūrus naują vartotojo giją, jai išskiriama ir nauja atmintinės sritis.

b. Daugiagijame vartotojo procese persijungiant nuo vienos gijos prie kitos OS visuomet
inicijuoja įprastą proceso konteksto perjungimą

c. Tiek vartotojo procesui, tiek vartotojo gijai sukurti reikalingi tokie patys sistemos ištekliai.

✓ Vartotojo procesą gali sudaryti daug vartotojo gijų

✓ Jei vartotojo procesas turi keletą gijų, gali iškilti poreikis šio proceso gijų veiksmus sinchronizuoti

✓ Vartotojo gijos kontekstą nusako programos skaitiklis (PC), registrų rinkinys ir dėklo erdvė

Iš 1-o žinių patikrinimo testo

Už ką yra atsakinga **išskirtinai tik** OS?

Select one or more:

- a. Prioriteto skirtingiems sistemos procesams priskyrimą
- b. Sąžiningą ir efektyvų turimų sistemos resursų paskirstymą
- c. Muzikos grojimą
- d. Sistemos naudotojo paskyrų kūrimą
- e. Tam tikro abstrakcijos lygio tarp proceso ir techninės įrangos užtikrinimą
- f. Procesų tarpusavio apsaugą

Iš 1-o žinių patikrinimo testo

Už ką yra atsakinga **išskirtinai tik** OS?

Select one or more:

- [ ] a. Prioriteto skirtingiems sistemos procesams priskyrimą
- [x] b. Sąžiningą ir efektyvų turimų sistemos resursų paskirstymą
- [ ] c. Muzikos grojimą
- [x] d. Sistemos naudotojo paskyrų kūrimą
- [x] e. Tam tikro abstrakcijos lygio tarp proceso ir techninės įrangos užtikrinimą
- [x] f. Procesų tarpusavio apsaugą

Iš 1-o žinių patikrinimo testo

Kokie mechanizmai reikalingi norint operacinėje sistemoje įgyvendinti multiprogramavimo (angl. *multiprogramming*) koncepciją? (Pažymėkite visus, kurie jūsų nuomone yra **<u>būtini</u>**)

Select one or more:

- ☐ a. Procesų planavimo mezhanizmas
- ☐ b. Sistemos laikrodžio mechanizmas
- ☐ c. Atmintinės valdymas
- ☐ d. Minimalus informavimo apie I/O pabaigą mechanizmas
- ☐ e. Laiko paskirstymo mechanizmas

Iš 1-o žinių patikrinimo testo

Kokie mechanizmai reikalingi norint operacinėje sistemoje įgyvendinti multiprogramavimo (angl. *multiprogramming*) koncepciją? (Pažymėkite visus, kurie jūsų nuomone yra **<u>būtini</u>**)

Select one or more:

- [x] a. Procesų planavimo mezhanizmas
- [ ] b. Sistemos laikrodžio mechanizmas
- [x] c. Atmintinės valdymas
- [x] d. Minimalus informavimo apie I/O pabaigą mechanizmas
- [ ] e. Laiko paskirstymo mechanizmas

Iš 1-o žinių patikrinimo testo

Nurodykite pagrindinį skirtumą tarp personaliniams kompiuteriams skirtų OS ir mainframe'ams skirtų OS:

Select one:

- a. Personaliniams kompiuteriams skirtose OS didžiausias dėmesys skiriamas tinklinių komunikacijų mechanizmo tobulinimui. Mainframe'ams skirtose OS daugiau koncentruojamasi efektyviam darbui su skaičiavimo procesais.
- b. Personaliniams kompiuteriams skirtose OS didžiausias dėmesys skiriamas sąsajos bei interaktyvumo su OS vartotoju tobulinimo klausimams. Mainframe'ams skirtose OS daugiau koncentruojamasi efektyviam darbui su skaičiavimo procesais.
- c. Personaliniams kompiuteriams skirtose OS didžiausias dėmesys skiriamas daugiaprocesio (angl. *multiprocessing*) darbo režimo tobulinimui. Mainframe'ams skirtose OS daugiau koncentruojamasi efektyviam darbui su skaičiavimo procesais ir daugiavartotojiškos aplinkos palaikymu.
- d. Personaliniams kompiuteriams skirtos OS ir mainframe'ams skirtos OS neturi esminių skirtumų.

Iš 1-o žinių patikrinimo testo

Nurodykite pagrindinį skirtumą tarp personaliniams kompiuteriams skirtų OS ir mainframe'ams skirtų OS:

Select one:

- a. Personaliniams kompiuteriams skirtose OS didžiausias dėmesys skiriamas tinklinių komunikacijų mechanizmo tobulinimui. Mainframe'ams skirtose OS daugiau koncentruojamasi efektyviam darbui su skaičiavimo procesais.
- b. Personaliniams kompiuteriams skirtose OS didžiausias dėmesys skiriamas sąsajos bei interaktyvumo su OS vartotoju tobulinimo klausimams. Mainframe'ams skirtose OS daugiau koncentruojamasi efektyviam darbui su skaičiavimo procesais.
- c. Personaliniams kompiuteriams skirtose OS didžiausias dėmesys skiriamas daugiaprocesio (angl. *multiprocessing*) darbo režimo tobulinimui. Mainframe'ams skirtose OS daugiau koncentruojamasi efektyviam darbui su skaičiavimo procesais ir daugiavartotojiškos aplinkos palaikymu.
- d. Personaliniams kompiuteriams skirtos OS ir mainframe'ams skirtos OS neturi esminių skirtumų.

Iš 1-o žinių patikrinimo testo

Kodėl vietoj tiesioginio sisteminių kreipinių vykdymo dažniausiai naudojamasi tam tikra taikomųjų programų sąsaja (API)? (Pažymėkite visus jūsų nuomone teisingus atsakymus)

Select one or more:

- a. API leidžia greičiau įvykdyti sisteminius kreipinius
- b. API užtikrina, kad sisteminių kreipinių parametrų reikšmės bus perduotos operacinei sistemai
- c. API užtikrina, kad techninės įrangos resursai bus paskirstyti. Tiesiogiai naudojant sisteminius kreipinius to užtikrinti neįmanoma
- d. API padeda lengviau realizuoti branduolio/vartotojo darbo režimo ideologiją
- e. API paslepia daugumą sisteminio kreipinio apdorojimo detalių

Iš 1-o žinių patikrinimo testo

Kodėl vietoj tiesioginio sisteminių kreipinių vykdymo dažniausiai naudojamasi tam tikra taikomųjų programų sąsaja (API)? (Pažymėkite visus jūsų nuomone teisingus atsakymus)

Select one or more:

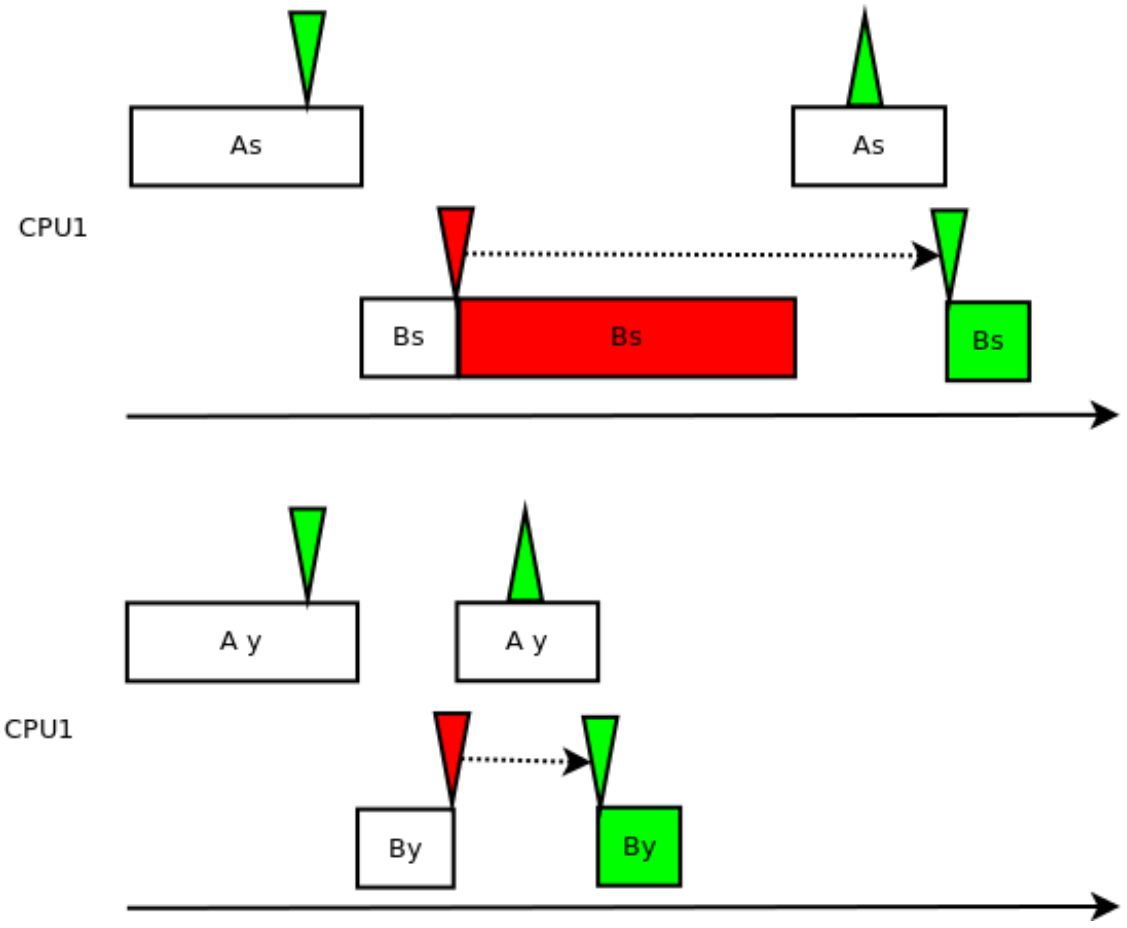
- a. API leidžia greičiau įvykdyti sisteminius kreipinius
- b. API užtikrina, kad sisteminių kreipinių parametrų reikšmės bus perduotos operacinei sistemai
- c. API užtikrina, kad techninės įrangos resursai bus paskirstyti. Tiesiogiai naudojant sisteminius kreipinius to užtikrinti neįmanoma
- d. API padeda lengviau realizuoti branduolio/vartotojo darbo režimo ideologiją
- e. API paslepia daugumą sisteminio kreipinio apdorojimo detalių

Atviri klausimai (1)

- Kada kritinės sekcijos sprendimams yra efektyvu naudoti aktyvaus laukimo užraktus (angl. spinlocks)? Kodėl?

Kodėl?

- 1 procesoriaus atvejis

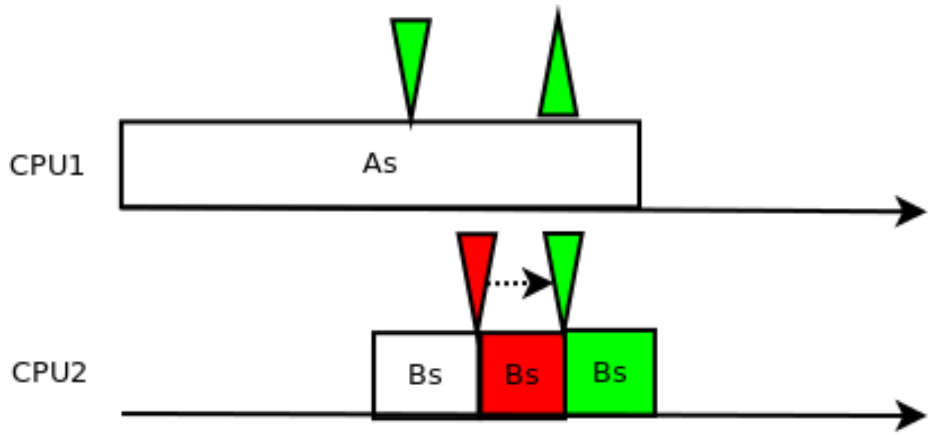


Konteksto perjung. - 3

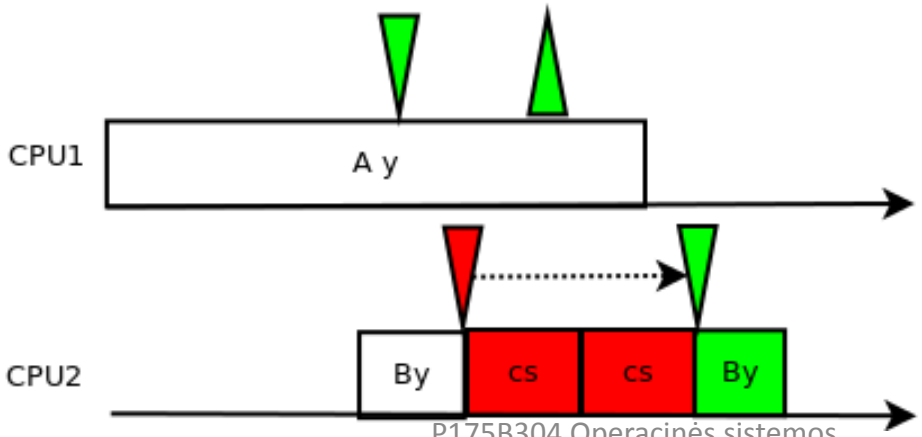
Konteksto perjung. - 3

Kodėl?

- kelių procesorių atvejis



Konteksto perjung. (CPU2) - 0



Konteksto perjung. (CPU2) - 2

Atviri klausimai (1)

- Kada kritinės sekcijos sprendimams yra efektyvu naudoti aktyvaus laukimo užraktus (angl. spinlocks)? Kodėl?
  - Kai turime daugiaprocesorę kompiuterinę sistemą (kai gijų kiekis grubiai atitinka ar yra mažesnis už procesorių skaičių)
  - ir kritinės sekcija yra nedidelė (naudojant blokavimą bus vykdomi du konteksto perjungimai, o tai mažina resursų naudojimo efektyvumą)

Atviri klausimai (2)

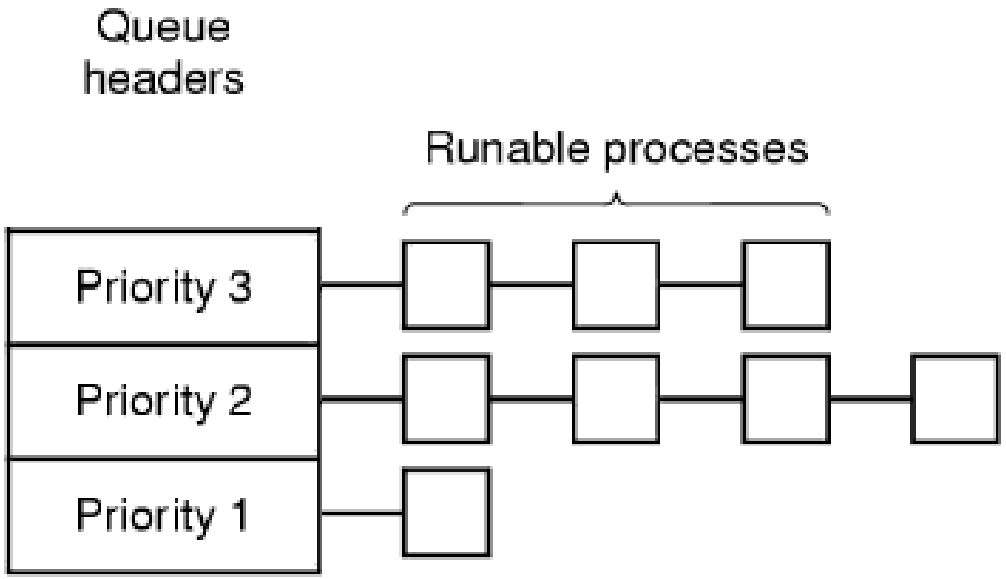
- Kaip sistemos administratorius jūs pastebėjote, kad naudojimosi sistemos resursais pikai yra nuo 10:00 iki 17:00 bei nuo 19:00 iki 22:00.
- Kompanijos valdyba nusprendė į jus kreiptis sistemos galinčios šiek tiek valdyti šią situaciją projektavimui.
- Pageidaujama piko valandomis išskirti trijų lygių vartotojus.
- Pirmojo lygio vartotojams reikalingi greitesni atsakymų laikai nei antro lygio vartotojams, o šiems savo ruožtu – geresni nei trečio lygio vartotojams.
- Jums reikia suprojektuoti tokią sistemą, kad visi vartotojai būtų aptarnaujami, tačiau išlaikant nurodytus reikalavimus.

Ar prioritetinis pertraukiamo tipo paprastas planavimas su 3-jų lygių fiksuoto prioriteto eilėmis išspręstų problemą?

Kodėl taip arba ne?

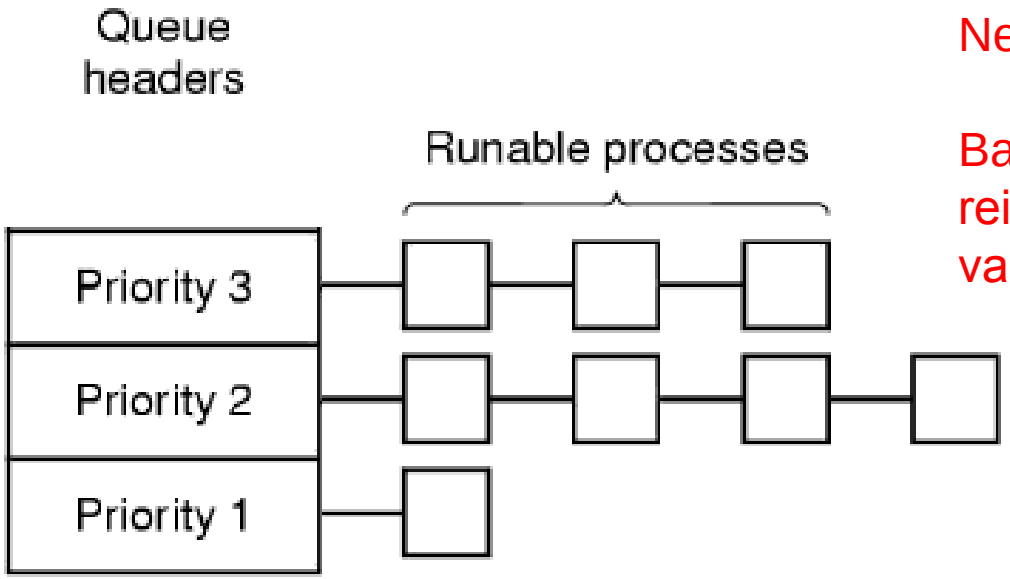
Prioritetinis planavimas (2)

- Prioritetai įvertina procesų svarbą
- Atsiradus pasiruošusių procesų eilėje procesui su didesniu prioritetu:
  - Proceso aptarnavimas gali būti ***nutraukiamas*** ir pradedamas vykdyti aukštesnio prioriteto procesas (pertraukiamo tipo)
  - Vykdomam procesui gali būti ***leidžiama užbaigti*** išnaudoti jam skirtą procesoriaus laiko kvantą ir tik po to pereinama prie proceso su aukštesniu prioritetu vykdymo (nepertraukiamo tipo)



Prioritetinis planavimas (2)

- Prioritetai įvertina procesų svarbą
- Atsiradus pasiruošusių procesų eilėje procesui su didesniu prioritetu:
  - Proceso aptarnavimas gali būti ***nutraukiamas*** ir pradedamas vykdyti aukštesnio prioriteto procesas (pertraukiamo tipo)
  - Vykdomam procesui gali būti ***leidžiama užbaigti*** išnaudoti jam skirtą procesoriaus laiko kvantą ir tik po to pereinama prie proceso su aukštesniu prioritetu vykdymo (nepertraukiamo tipo)



Netinka:

Badavimo situacija (neatitinka reikalavimo užtikrinti visų vartotojų aptarnavimą)

Procesų sinchronizacijos problema

Procesų sinchronizacijos problemų aprašymui panaudota žemiau pateikta gyvenimiška situacija:

Jūs su savo draugu dirbate to paties viešbučio administratoriais. Paprastai dirbate poroje (t.y. dviese vienu metu). Viešbučio politika sako, kad viešbučio svečių registracijos erdvėje visuomet būtų pamerkta vaza gėlių. Su savo kolega susitariate, kad, norėdami įgyvendinti viešbučio politiką, kiekvienas tik atėjęs į darbą atkreipsite dėmesį į šią erdvę. Pastebėję, kad nėra gėlių, tuoj pat vykstate į artimiausią gėlių parduotuvę jų nupirkti.

Kurią iš procesų sinchronizacijos problemų aprašo aukščiau pateiktas scenarijus?

Procesų sinchronizacijos problema

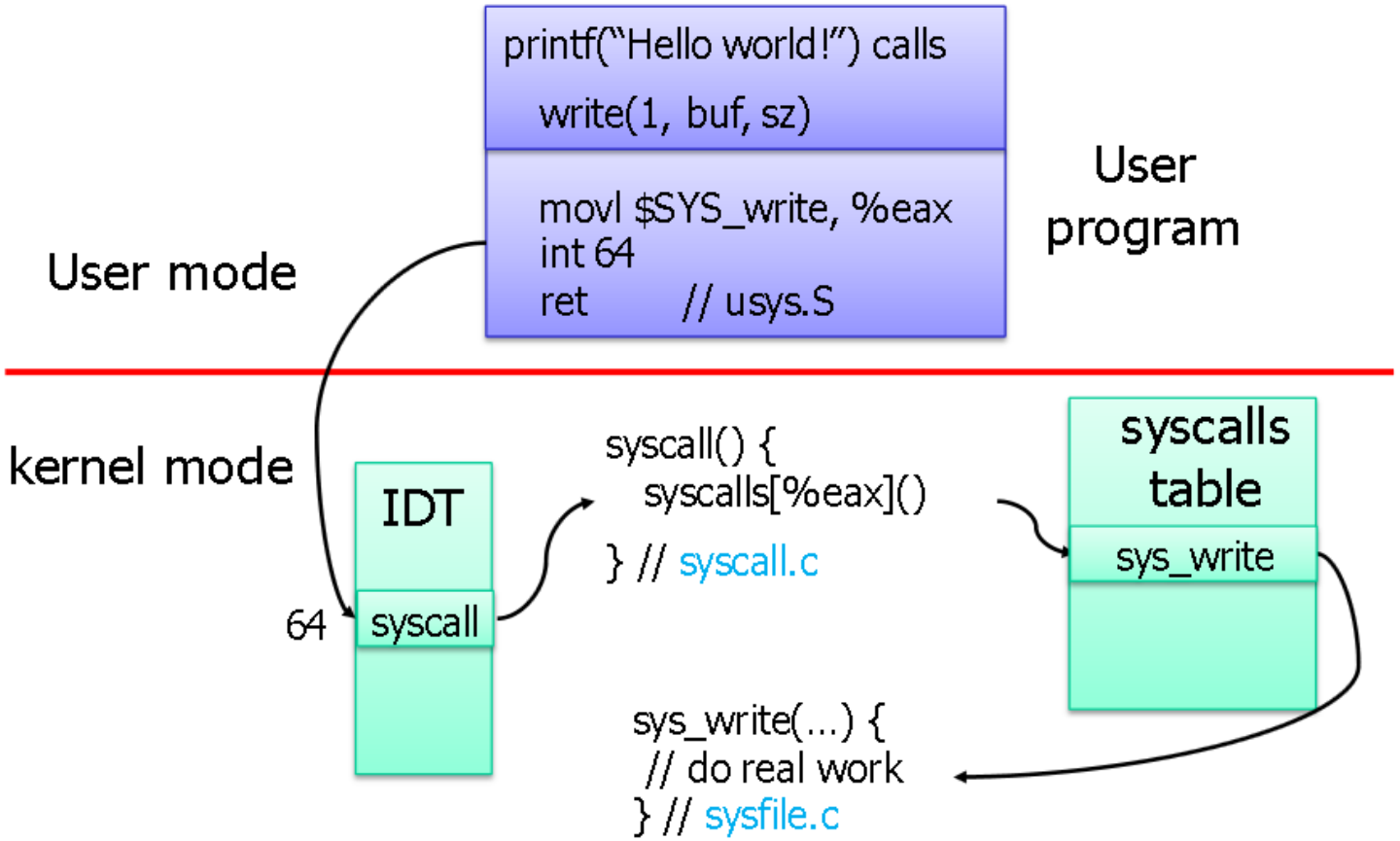
Procesų sinchronizacijos problemų aprašymui panaudota žemiau pateikta gyvenimiška situacija:

Jūs su savo draugu dirbate to paties viešbučio administratoriais. Paprastai dirbate poroje (t.y. dviese vienu metu). Viešbučio politika sako, kad viešbučio svečių registracijos erdvėje visuomet būtų pamerkta vaza gėlių. Su savo kolega susitariate, kad, norėdami įgyvendinti viešbučio politiką, kiekvienas tik atėjęs į darbą atkreipsite dėmesį į šią erdvę. Pastebėję, kad nėra gėlių, tuoj pat vykstate į artimiausią gėlių parduotuvę jų nupirkti.

Kurią iš procesų sinchronizacijos problemų aprašo aukščiau pateiktas scenarijus?

Lenktynių situacija (angl. race condition) – situacija, kai rezultatas priklauso nuo procesų (gijų) vykdymo sekos.

OS paslaugos/pertrauktys/sisteminiai kreipiniai:

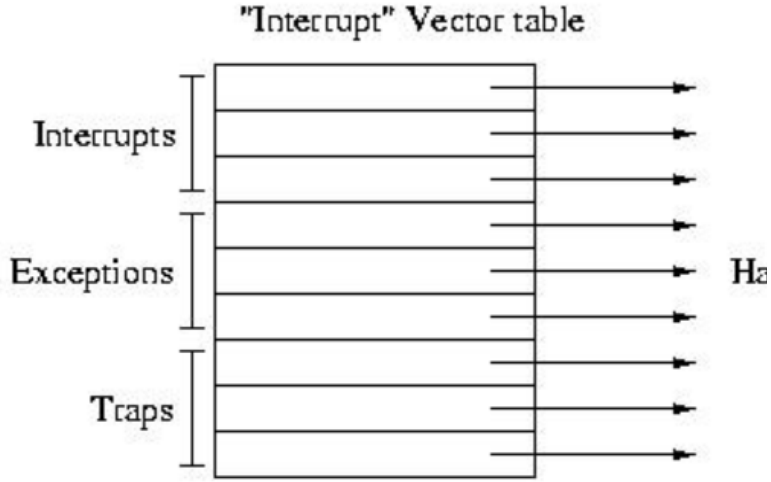


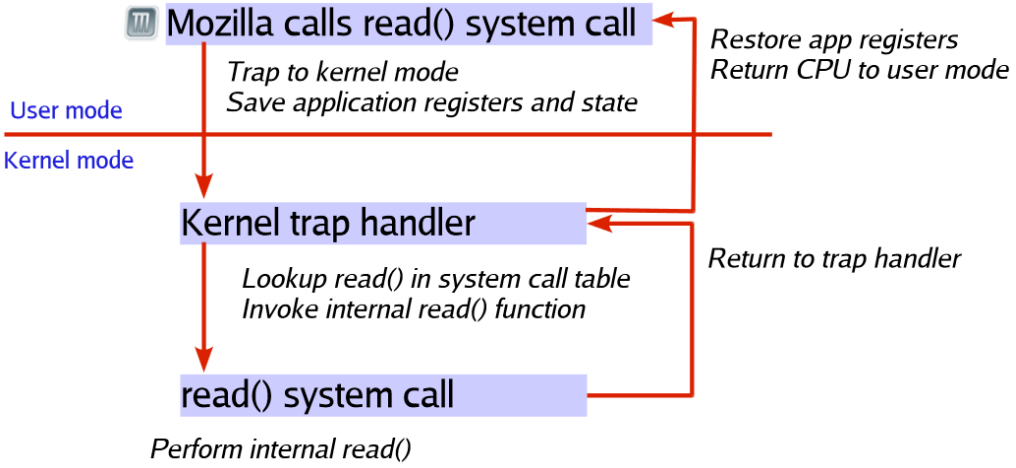
Paveikslėlio aprašymas

- Vartotojo programa iškviečia funkciją printf su parametru "Hello world!". Printf suformatuoja (print formatted) išvedamą stringą ir iškviečia write su parametrais 1 - fd į kurį bus rašomas tekstas, buf - spausdinamas buferis, sz - spausdinamo buferio ilgis.
- Funkcija write yra įgyvendinta asemblerio kodu. Į registrą eax įrašoma reikšmė SYS_write ir panaudojamas 64'as interruptas, kuris perduoda kontrolę kerneliui. Kai kernelis baigs darbą, bus įvykdoma likusi ret instrukcija.
- Kernelis gavęs interruptą pasižiūri į savo "Interrupt Descriptor Table" ir radęs ten 64'a reikšmę supranta, kad pašaukusi funkcija nori atlikti system call'ą (64 interruptas reiškia syscall).
- syscall.c yra aprašyta kernelio funkciją syscall, kuri pašaukia būtent toki syscall'ą, kuris yra įrašytas eax registre. syscalls[eax] translate'inasi į sys_write() funkcija.
- sys_write() iš steko pasiims buferio dydį, buferį (adresą?) ir target fd ir išspausdins tekstą į duotą fd.

OS paslaugos/pertrauktys/sisteminiai kreipiniai:

Sisteminio kreipinio apdorojimas operacinėje sistemoje skiriasi/nesiskiria nuo x86 architektūros aparatinės pertraukties apdorojimo. Jei skiriasi, įvardinkite kas (ne daugiau 70 simbolių)

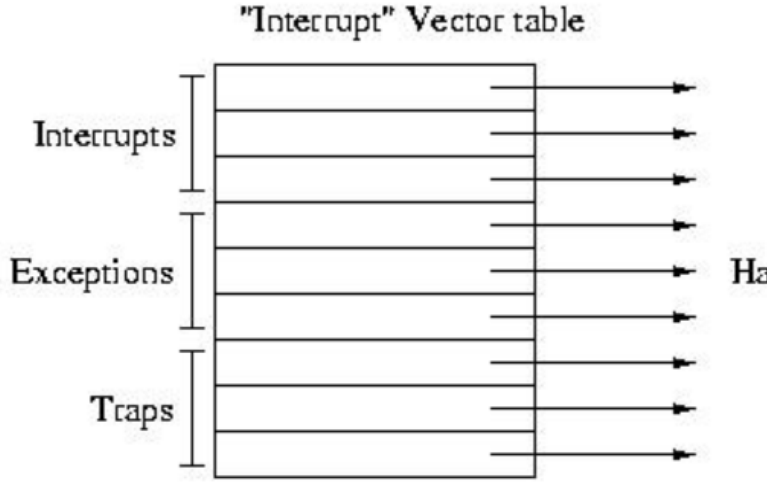


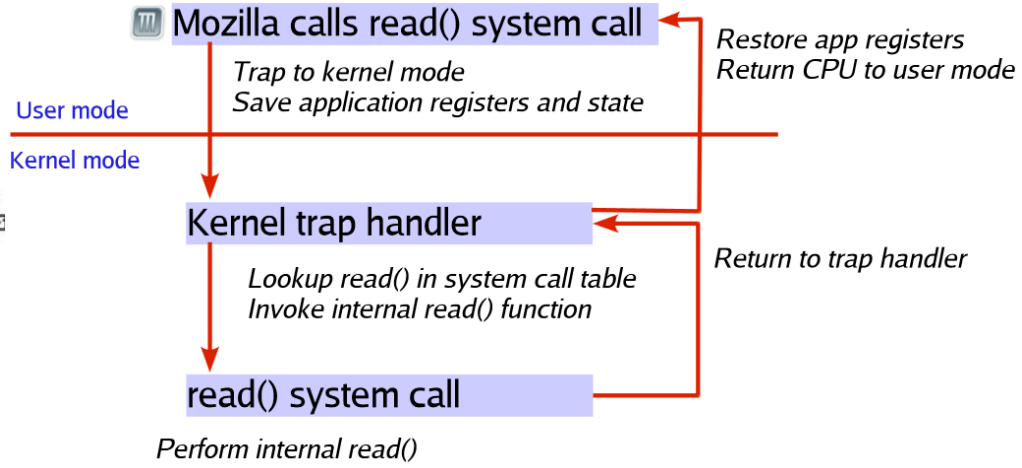


Interrupt: Asynchronous to the CPU
Exception: Something out of the ordinary
Trap: Invoked explicitly to cause a particular behavior

OS paslaugos/pertrauktys/sisteminiai kreipiniai:

Sisteminio kreipinio apdorojimas operacinėje sistemoje skiriasi/nesiskiria nuo x86 architektūros aparatinės pertraukties apdorojimo. Jei skiriasi, įvardinkite kas (ne daugiau 70 simbolių)





Interrupt: Asynchronous to the CPU
Exception: Something out of the ordinary
Trap: Invoked explicitly to cause a particular behavior

Skaičiuojamojo tipo klausimai (1)

Jei reikalinga atlikite skaičiavimus ir atsakykite į žemiau pateiktus klausimus

- Pasiruošusių procesų eilėje turime 5 procesus. Prognozuojami jų įvykdymui reikalingi laikai yra 9, 6, 3, 5 ir X (X kinta nuo 1 iki 100). Kokia tvarka jie turėtų būti įvykdyti, kad vidutinis atsakymo laikas būtų minimalus? Ar galėtumėte įvardinti planavimo algoritmą (-us), kuris (-ie) galėtų būtų panaudoti tokios procesų vykdymo sekos realizacijai?

Skaičiuojamojo tipo klausimai (1)

Jei reikalinga atlikite skaičiavimus ir atsakykite į žemiau pateiktus klausimus

- Pasiruošusių procesų eilėje turime 5 procesus. Prognozuojami jų įvykdymui reikalingi laikai yra 9, 6, 3, 5 ir X (X kinta nuo 1 iki 100). Kokia tvarka jie turėtų būti įvykdyti, kad vidutinis atsakymo laikas būtų minimalus? Ar galėtumėte įvardinti planavimo algoritmą (-us), kuris (-ie) galėtų būtų panaudoti tokios procesų vykdymo sekos realizacijai?

$0<X\leq3$:	X,3,5,6,9
$3<X\leq5$:	3,X,5,6,9
$5<X\leq6$:	3, 5,X,6,9
$6<X\leq9$:	3, 5, 6,X,9
$X>9$:	3, 5, 6, 9,X.

Planavimo algoritmas:
SPF (SJF) – trumpiausias procesas pirmas

Skaičiuojamojo tipo klausimai (2)

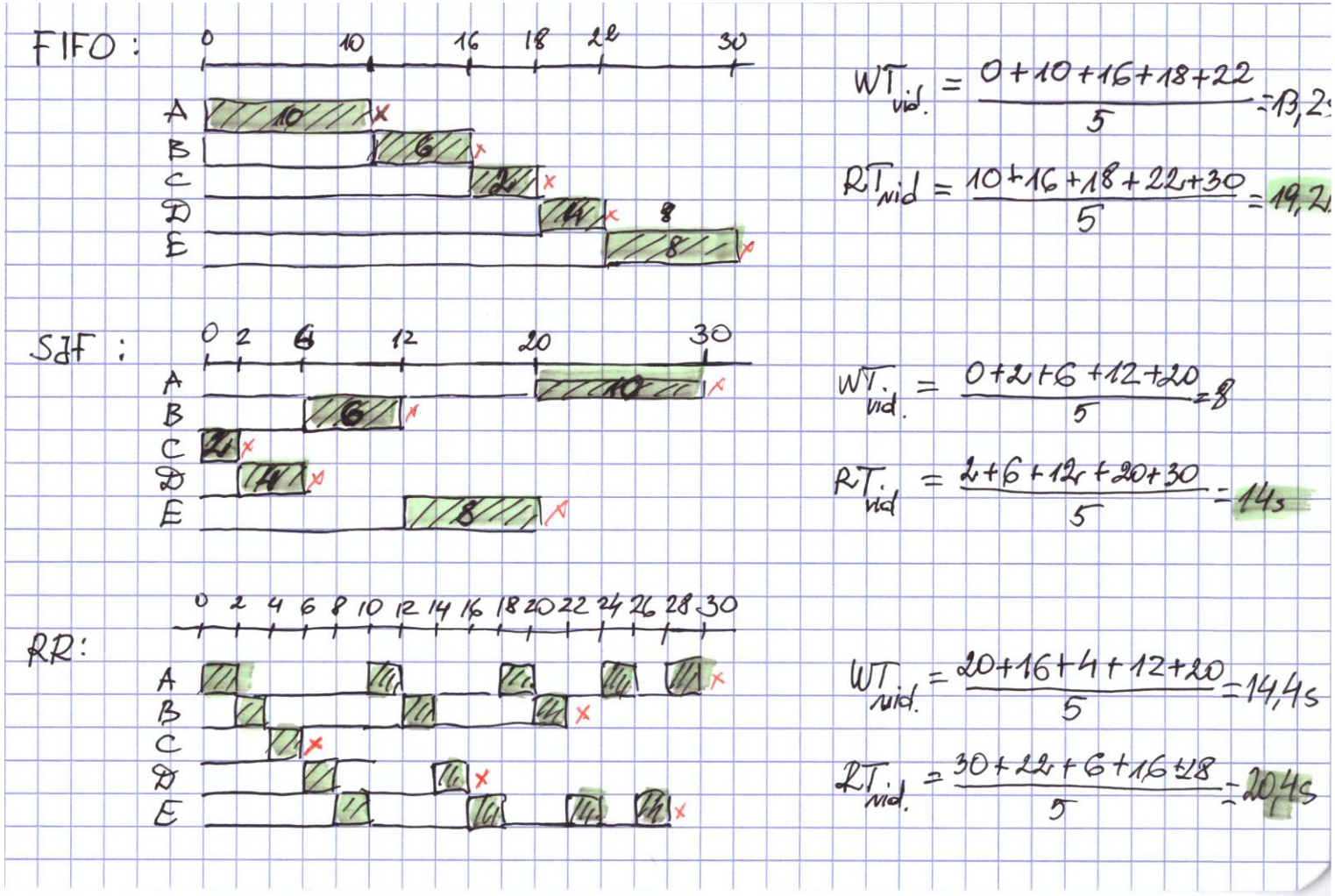
- Penki skaičiavimo tipo procesai paleidžiami vienu metu. Atskirų procesų vykdymo laikai atitinkamai yra 10, 6, 2, 4 ir 8 sekundės.
- Koks bus vidutinis procesų įvykdymo laikas (angl. turnaround time) esant 1 procesoriui, kai planavimui naudojami sekantys planavimo algoritmai:
  - Ciklinis aptarnavimas (kai laiko kvantas – 2s)
  - FCFS
  - SJF

Skaičiuojamojo tipo klausimai (2)

- Penki skaičiavimo tipo procesai paleidžiami vienu metu. Atskirų procesų vykdymo laikai atitinkamai yra 10, 6, 2, 4 ir 8 sekundės.
- Koks bus vidutinis procesų įvykdymo laikas (angl. turnaround time) esant 1 procesoriui, kai planavimui naudojami sekantys planavimo algoritmai:
  - Ciklinis aptarnavimas (kai laiko kvantas – 2s) - **20,4 s**
  - FCFS - **19,2 s**
  - SJF - **14 s**

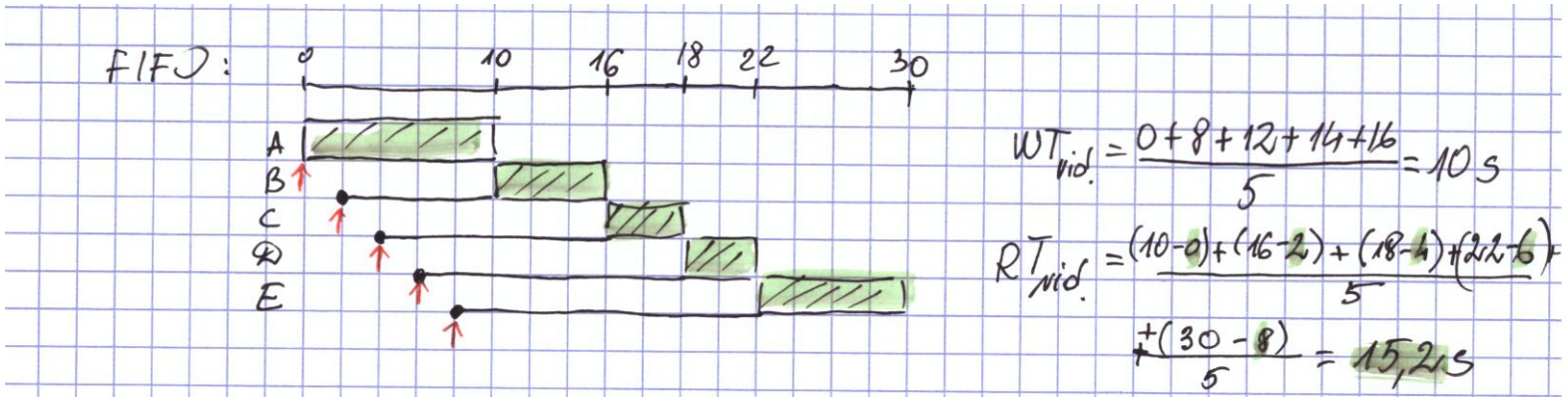
Skaičiuojamojo tipo klausimai (2)

- Kai procesai pasirodo vienu metu



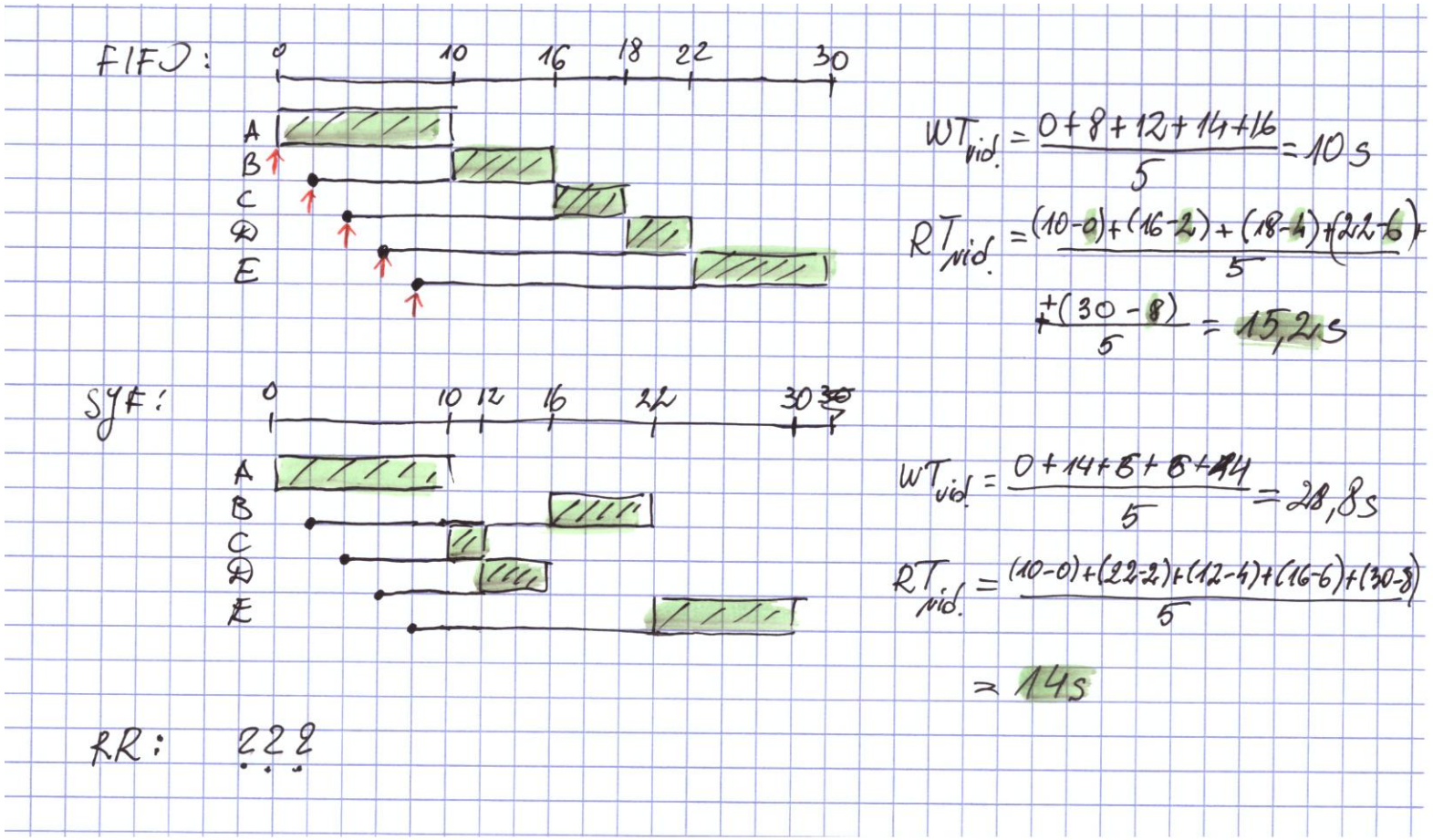
Skaičiuojamojo tipo klausimai (2)

- Kai procesai pasirodo skirtingais laiko momentais



Skaičiuojamojo tipo klausimai (2)

- Kai procesai pasirodo skirtingais laiko momentais



Analizės tipo klausimai (1)

Išanalizuokite ir atsakykite

- Išanalizuokite žemiau pateikto vartotojo-gamintojo algoritmo pseudo kodą ir atsakykite į sekančius klausimus:
  - Ar šiame algoritme naudojamas baigtinio ar neriboto dydžio buferis? Atsakymą pagrįskite
  - Ar šis algoritmas veiks korektiškai esant tokiai komandų sekai, kokia nurodyta pateiktame pavyzdyje. Atsakymą pagrįskite

Vartotojo-gamintojo uždavinys

```
0     // Pradinės reikšmės
1     S.count:=1;
2     N.count:=0;
3     in:=out:=0;
4
5     // F-ja kviečiama iš Gamintojas (Producer)
6     append(v):
7     b[in]:=v;
8     in++;
9
10    // F-ja kviečiama iš Vartotojas (Consumer)
11    take():
12    w:=b[out];
13    out++;
14    return w;
15
16    // Gamintojas (Producer)
17    Gamintojas (Producer):
18    Repeat
19      Gaminti v;
20      wait (S);
21      append (v); /* kritinė sekcija */
22      signal (S);
23      signal (N);
24    forever
```

```
25 // Vartotojas (Consumer)
26    Vartotojas (Consumer):
27    Repeat
28      wait (S);
29      wait (N);
30      w:=take ();/* kritinė sekcija */
31      signal (S);
32      vartoti (w);
33    forever
```

Vartotojo-gamintojo uždavinys

```
0     // Pradinės reikšmės
1     S.count:=1;
2     N.count:=0;
3     in:=out:=0;
4
5     // F-ja kviečiama iš Gamintojas (Producer)
6     append(v):
7     b[in]:=v;
8     in++;
9
10    // F-ja kviečiama iš Vartotojas (Consumer)
11    take():
12    w:=b[out];
13    out++;
14    return w;
15
16    // Gamintojas (Producer)
17    Gamintojas (Producer):
18    Repeat
19      Gaminti v;
20      wait (S);
21      append (v); /* kritinė sekcija */
22      signal (S);
23      signal (N);
24    forever
```

Dvejetainis sem. Naudojamas prieigos prie buferio užtikrinimui

Skaitmeninis sem. Pradžioje gaminių nėra. Patikrinimui naudojamas wait(N)

1.

2.

```
25 // Vartotojas (Consumer)
26    Vartotojas (Consumer):
27    Repeat
28      wait (S);
29      wait (N);
30      w:=take ();/* kritinė sekcija */
31      signal (S);
32      vartoti (w);
33    forever
```

Vartotojo-gamintojo uždavinys

```
0     // Pradinės reikšmės
1     S.count:=1;
2     N.count:=0;
3     in:=out:=0;
4
5     // F-ja kviečiama iš Gamintojas (Producer)
6     append(v):
7     b[in]:=v;
8     in++;
9
10    // F-ja kviečiama iš Vartotojas (Consumer)
11    take():
12    w:=b[out];
13    out++;
14    return w;
15
16    // Gamintojas (Producer)
17    Gamintojas (Producer):
18    Repeat
19      Gaminti v;
20      wait (S);
21      append (v); /* kritinė sekcija */
22      signal (S);
23      signal (N);
24    forever
```

Dvejetainis sem. Naudojamas prieigos prie buferio užtikrinimui

Skaitmeninis sem. Pradžioje gaminių nėra. Patikrinimui naudojamas wait(N)

```
25 // Vartotojas (Consumer)
26    Vartotojas (Consumer):
27    Repeat
28      wait (S);
29      wait (N);
30      w:=take ();/* kritinė sekcija */
31      signal (S);
32      vartoti (w);
33    forever
```

Vartotojo-gamintojo uždavinys

```
0     // Pradinės reikšmės
1     S.count:=1;
2     N.count:=0;
3     in:=out:=0;
4
5     // F-ja kviečiama iš Gamintojas (Producer)
6     append(v):
7     b[in]:=v;
8     in++;
9
10    // F-ja kviečiama iš Vartotojas (Consumer)
11    take():
12    w:=b[out];
13    out++;
14    return w;
15
16    // Gamintojas (Producer)
17    Gamintojas (Producer):
18    Repeat
19      Gaminti v;
20      wait (S);
21      append (v); /* kritinė sekcija */
22      signal (S);
23      signal (N);
24    forever
```

Dvejetainis sem. Naudojamas prieigos prie buferio užtikrinimui

Skaitmeninis sem. Pradžioje gaminių nėra. Patikrinimui naudojamas wait(N)

1.

2.

```
25 // Vartotojas (Consumer)
26    Vartotojas (Consumer):
27    Repeat
28      wait (S);
29      wait (N);
30      w:=take ();/* kritinė sekcija */
31      signal (S);
32      vartoti (w);
33    forever
```

Analizės tipo klausimai (1)

Išanalizuokite ir atsakykite

- Ar šiame algoritme naudojamas baigtinio ar neriboto dydžio buferis? Atsakymą pagrįskite

Neriboto dydžio nes nėra semaforo, kuriuo būtų sinchronizuojami vartotojo ir gamintojo veiksmai laisvų vietų skaičiaus buferyje atžvilgiu (pagr. šalt.179-180 psl.)

Analizės tipo klausimai (1)

Išanalizuokite ir atsakykite

– Ar šis algoritmas veiks korektiškai esant tokiai komandų sekai, kokia nurodyta pateiktame pavyzdyje. Atsakymą pagrįskite

Esant tokiai komandų sekai kokia pateikta šiame pavyzdyje, vartotojo procesas gali patekti į mirties tašką – įeiti į kritinę sekciją esant tuščiam buferiui ir negalėti nei imti iš buferio informacijos, nei atlaisvinti semaforo S (žr. pagr. šalt. 179 psl). Kad algoritmas veiktų korektiškai reikia vartotojo programos tekste 28 ir 29 eilutes sukeisti vietomis.

Analizės tipo klausimai (2)

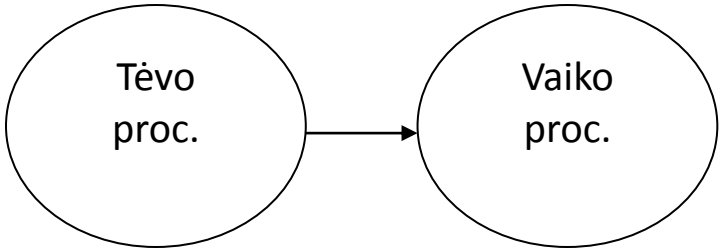
- Išanalizuokite žemiau pateiktą programos išeities tekstą ir schematiškai atvaizduokite kiek procesų ir kokia jų hierarchija bus sukurta.

```
1       #include <sys/types.h>
2       #include <stdio.h>
3       #include <unistd.h>
4       int main ()
5       {
6       pid_t pid;
7       pid = fork ();
```

Analizės tipo klausimai (2)

- Išanalizuokite žemiau pateiktą programos išeities tekstą ir schematiškai atvaizduokite kiek procesų ir kokia jų hierarchija bus sukurta.

```
1       #include <sys/types.h>
2       #include <stdio.h>
3       #include <unistd.h>
4       int main ()
5       {
6       pid_t pid;
7       pid = fork ();
```



Analizės tipo klausimai (3)

- Sekančioje skaidrėje pateiktas Soliario OS TS klasės dispečerio konfigūracijos failas. Išanalizavę jo turinį atsakykite į sekančius klausimus (S):
  - Koks procesų planavimo mechanizmas naudojamas planuojant šiai klasei priskiriamų procesų vykdymą?
  - Ar šiame mechanizme naudojama tokia abtrakcija kaip proceso prioritetas?

Analizės tipo klausimai (3)

TS prioritetinės klasės dispečerio konfigūracija

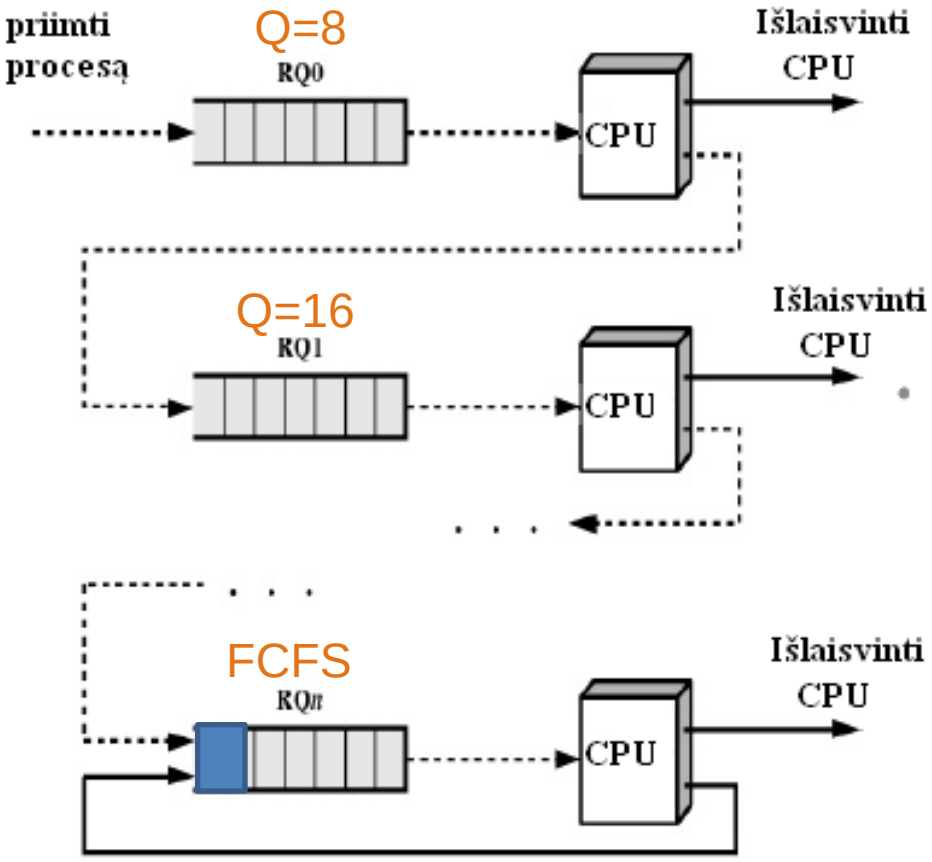
```
-bash-3.00$ dispadmin -c TS -g
# Time Sharing Dispatcher Configuration
RES=1000

# ts_quantum ts_tqexp  ts_slpret  ts_maxwait ts_lwait  PR- LEVEL
200           0          50           0        50          #      0
200           0          50           0        50          #      1
200           0          50           0        50          #      2
.....
```

- ***ts_quantum***: tai laikas pagal nutylėjimą skirtas tam tikram prioritetui.
- ***ts_tqexp:*** tai naujas prioritetas procesui , kuris pilnai išnaudojo laiko kvantą.
- ***ts_slpret:*** naujas prioritetas procesui, kuris užsiblokavo neišnaudojęs laiko kvanto.
- ***ts_maxwait***: jei gija negauna CPU per laiką ts_maxwait, jos prioritetas paaukštinamas iki ts_lwait.
- ***ts_lwait***.

Daugiau info: `$man ts_dptbl`

MLFQ (3)



Pagr. veikimo principai:

- Pirmą kartą procesas talpinamas į aukščiausio prioriteto eilę.
- Jei išnaudotas CPU laiko kvantas, bet procesas nepasibaigė – perkelti jį į žemesnio prioriteto eilę
- Jei procesas neišnaudojo jam skirto laiko kvanto – perkelti jį į aukštesnio prioriteto eilę arba palikti esamoje.

Analizės tipo klausimai (4)

- Išanalizuokite sekančioje skaidrėje pateiktą programos išeities tekstą, kuriame pateiktas filosofų problemos sprendimas. Kodėl būsenos kintamasis state [i] yra nustatomas į HUNGRY reikšmę procedūroje take_forks ()? Atsakymą pagrįskite.

```
#define N            5              /* number of philosophers */
#define LEFT         (i+N-1)%N      /* number of i's left neighbor */
#define RIGHT        (i+1)%N        /* number of i's right neighbor */
#define THINKING     0              /* philosopher is thinking */
#define HUNGRY       1              /* philosopher is trying to get forks */
#define EATING       2              /* philosopher is eating */
typedef int semaphore;              /* semaphores are a special kind of int */
int state[N];                       /* array to keep track of everyone's state */
semaphore mutex = 1;                /* mutual exclusion for critical regions */
semaphore s[N];                     /* one semaphore per philosopher */

void philosopher(int i)             /* i: philosopher number, from 0 to N1 */
{
     while (TRUE){                  /* repeat forever */
          think();                  /* philosopher is thinking */
          take_forks(i);            /* acquire two forks or block */
          eat();                    /* yum-yum, spaghetti */
          put_forks(i);             /* put both forks back on table */
     }
}

void take_forks(int i)              /* i: philosopher number, from 0 to N1 */
{
     down(&mutex);                  /* enter critical region */
     state[i] = HUNGRY;             /* record fact that philosopher i is hungry */
     test(i);                       /* try to acquire 2 forks */
     up(&mutex);                    /* exit critical region */
     down(&s[i]);                   /* block if forks were not acquired */
}

void put_forks(i)                   /* i: philosopher number, from 0 to N1 */
{
     down(&mutex);                  /* enter critical region */
     state[i] = THINKING;           /* philosopher has finished eating */
     test(LEFT);                    /* see if left neighbor can now eat */
     test(RIGHT);                   /* see if right neighbor can now eat */
     up(&mutex);                    /* exit critical region */
}

void test(i)                        /* i: philosopher number, from 0 to N1* /
{
     if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING) {
          state[i] = EATING;
          up(&s[i]);
```

Analizės tipo klausimai (4)

- Išanalizuokite pateiktą programos išeities tekstą, kuriame pateiktas filosofų problemos sprendimas. Kodėl būsenos kintamasis state [i] yra nustatomas į HUNGRY reikšmę procedūroje take_forks ()? Atsakymą pagrįskite.

- Jei filosofo procesas blokuojasi, kaimynai filosofai gali vėliau nustatyti, kad jis reikalavo resursų (bet negavo) t.y. yra alkanas (HUNGRY). Vykdant test procedūrą šis statusas procesams kaimynams tikrinamas ir jei būsena atitinka alkano proceso (HUNGRY), tai kaimynas filosofas žadinimas ir jo būsena keičiama į EATING.