

Operacinės sistemos

P175B304

03T

2018-02-19

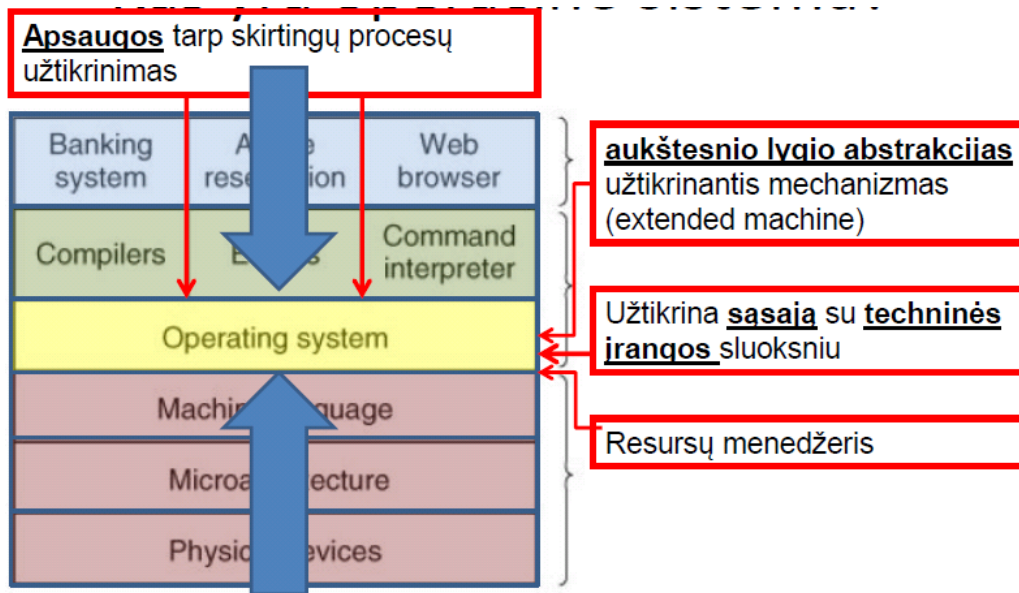
2018-02-20

Praėjusį kartą paskaitos metu:

- OS apibrėžimas
- OS evoliucija (trumpai)

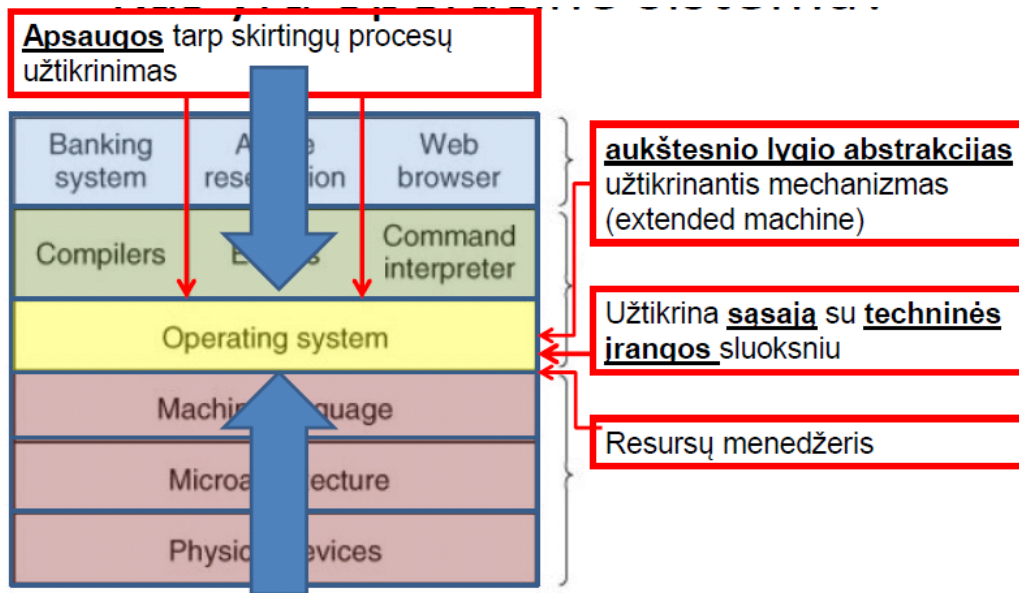
Praėjusį kartą paskaitos metu:

- OS apibrėžimas
- OS evoliucija (trumpai)



Praėjusį kartą paskaitos metu:

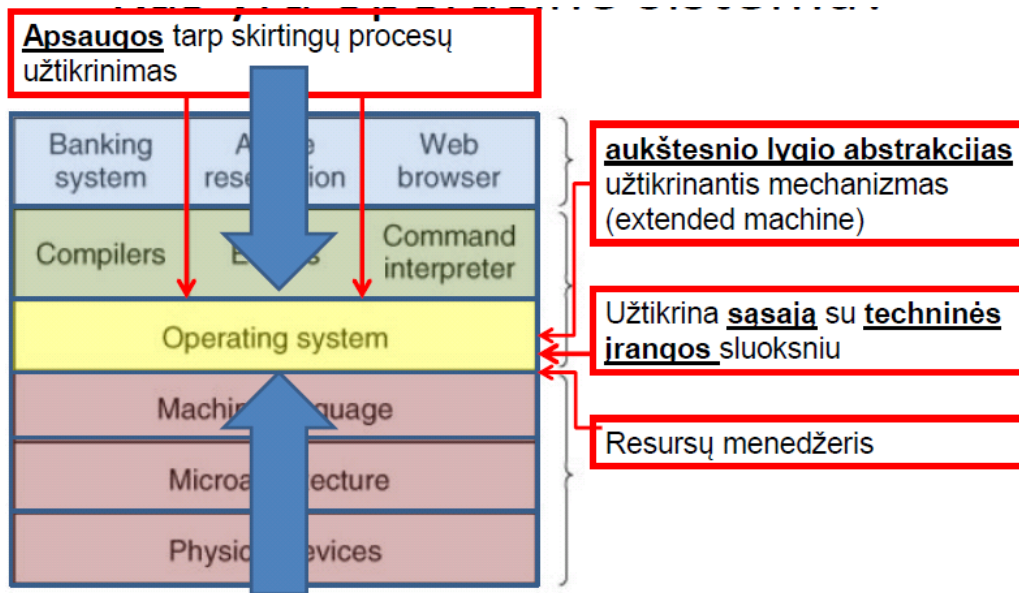
- OS apibrėžimas
- OS evoliucija (trumpai)



- Leisti abstrakcijomis manipuluoti
- Paslėpti techninės įrangos detales
- Abstrakcijas apsaugoti

Praėjusį kartą paskaitos metu:

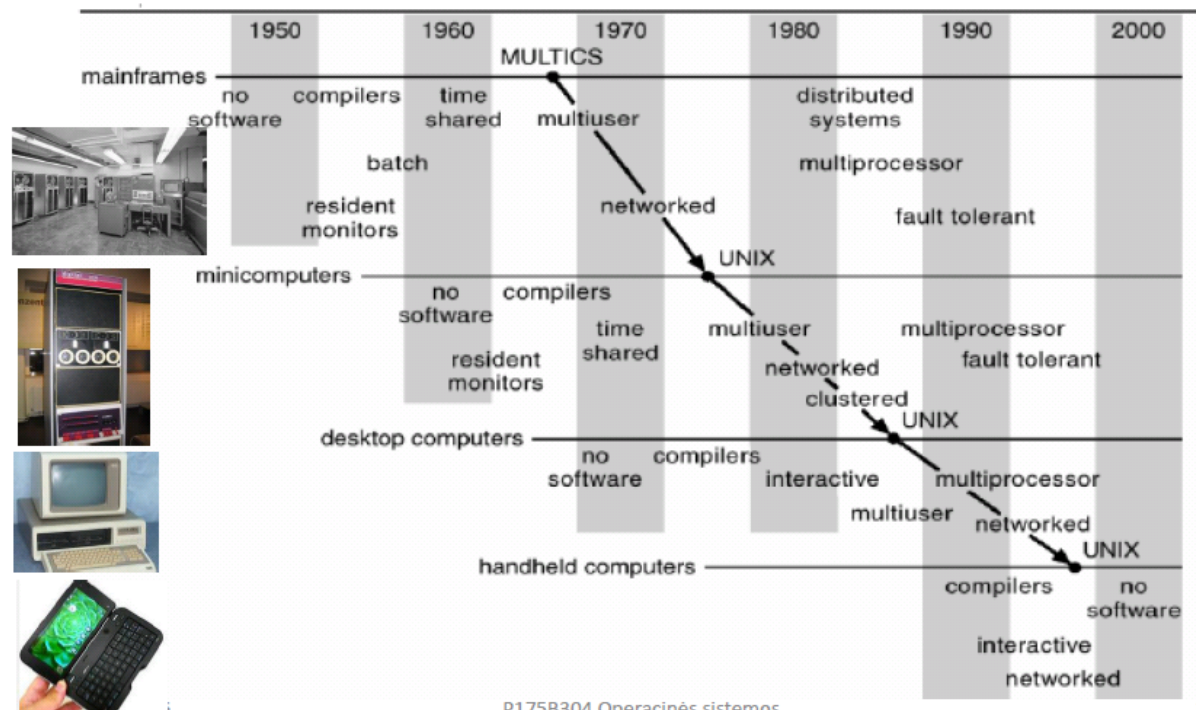
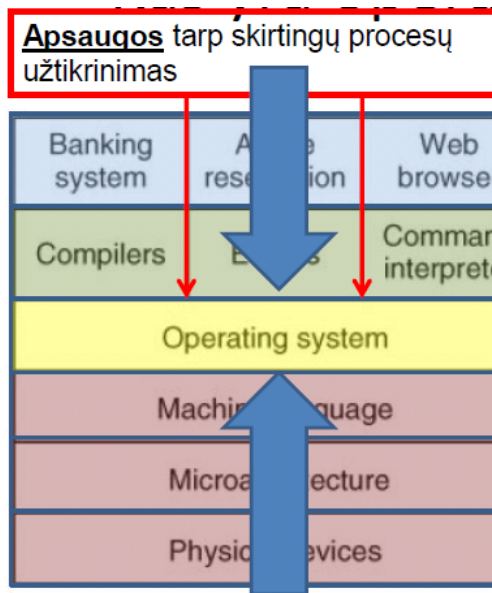
- OS apibrėžimas
- OS evoliucija (trumpai)



- Leisti abstrakcijomis manipuluoti
- Paslėpti techninės įrangos detales
- Abstrakcijas apsaugoti
- Užtikrinti efektyvų visos kompiuterinės sistemos resursų panaudojimą.

Praėjusį kartą paskaitos metu:

- OS apibrėžimas
- OS evoliucija (trumpai)



P175B304 Operacinės sistemos

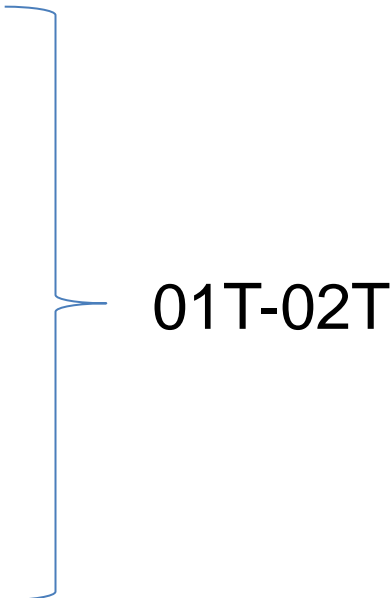
Savarankiškai:

- OS apibrėžimas
- OS evoliucija (trumpai)
- OS evoliucija (išsamiau)
- OS architektūros
- OS paslaugos, sisteminiai kreipiniai



01T-02T

Savarankiškai:

- OS apibrėžimas
 - OS evoliucija (trumpai)
 - OS evoliucija (išsamiau)
 - OS architektūros
 - OS paslaugos, sisteminiai kreipiniai
- 
- 01T-02T
- kontaktinių susitikimų metu plačiau aptarsime **tik tam tikrus** arba **svarbesnius** teorinius aspektus

Teorinių žinių testo patikrinimo testas

- Aktyvus iki semestro pabaigos. Galima atlikti keletą kartų.
- Skirtas padėti jums įsisavinti medžiagą (ieškodami atsakymų į pateiktus klausimus atkreipsite dėmesį į tuos dalykus, kurie vienoje ar kitoje temoje yra svarbūs).
- Žinių patikrinimo testo įvertis tik jums – papildomi balai už šią veiklą neskaičiuojami.

Paskaitos turinys

- Procesų valdymas
 - **Procesas. Jo būvis, kontekstas. Persijungimas nuo vieno proceso prie kito.**
 - Gijos, realizacijos modeliai. Proceso-gijos skirtumai.
 - Tarprocesinė (IPC) komunikacija, klasikinės IPC komunikacijos problemos.
 - Procesų vykdymo planavimas.

Kas yra procesas?

- Įvadinis testas...

Procesas tai (pažymėkite labiausiai tinkamą atsakymą):



Sisteminis kreipinys (OS funkcija)



Atminties adresų erdvė bei papildomai viena ar kelios gijos



Tai programa saugoma pastovioje atmintyje (diske, CD/DVD, flash ir t.t.)



Techninės įrangos abstrakcija

Kas yra procesas?

Modelio atsakas	Dalinis kreditas	Skaičius	Dažnis
Techninės įrangos abstrakcija	0.00%	20	5.21%
Tai programa saugoma pastovioje atmintyje (diske, CD/DVD, flash ir t.t.)	0.00%	11	2.86%
Atminties adresų erdvė bei papildomai viena ar kelios gijos	100.00%	88	22.92%
Sisteminis kreipinys (operacinės sistemos funkcija)	0.00%	257	66.93%
[Nėra atsako]	0.00%	8	2.08%

Procesas iš sistemos vartotojo perspektyvos

Procesas iš sistemos vartotojo perspektyvos

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

```
inglagz@os:~$ ps -f -u inglagz | sort +7
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
inglagz	3738	8444	0	11:38:03	pts/18	0:00	nano f2.txt
inglagz	3739	8444	0	11:38:03	pts/18	0:00	nano f3.txt
inglagz	3740	8444	0	11:38:03	pts/18	0:00	nano list.txt
inglagz	3741	8444	0	11:38:03	pts/18	0:00	nano prog1.c
inglagz	3742	8444	0	11:38:03	pts/18	0:00	nano sbin.txt

```
inglagz@os:~$
```

Procesas iš OS perspektyvos

- 3 pagrindiniai komponentai:

Procesas iš OS perspektyvos

- 3 pagrindiniai komponentai:
 - Proceso adresų erdvė
 - op. atminties dalis, kuri yra pasiekama procesui
 - Skirta įvairių komponentų (programos teksto, kintamųjų, steko, etc.) saugojimui

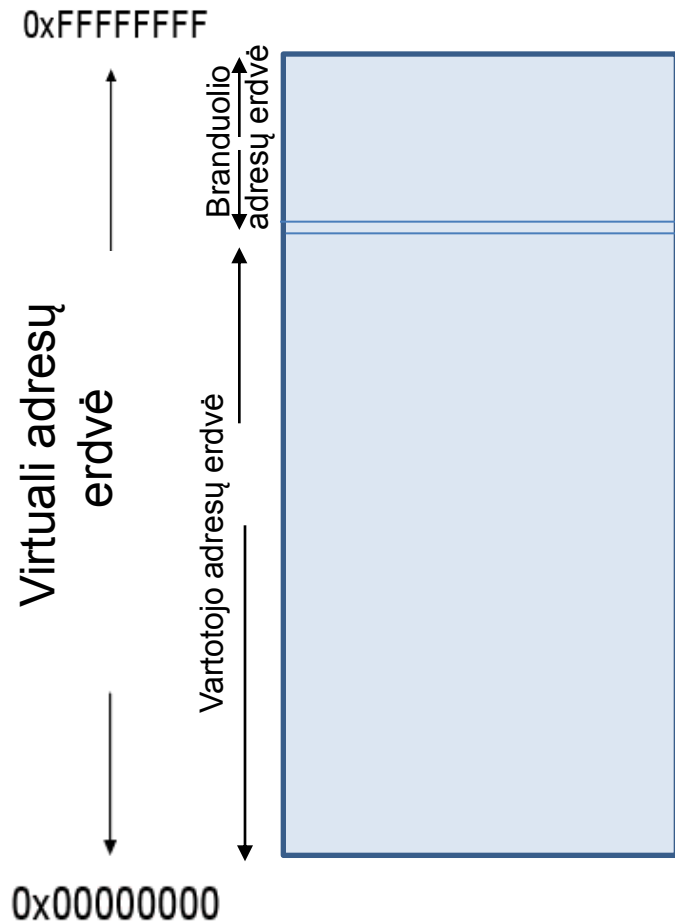
Procesas iš OS perspektyvos

- 3 pagrindiniai komponentai:
 - Proceso adresų erdvė
 - op. atminties dalis, kuri yra pasiekama procesui
 - Skirta įvairių komponentų (programos teksto, kintamųjų, steko, etc.) saugojimui
 - Procesoriaus (CPU) būseną
 - Su programos vykdymu susiję CPU registrai
 - Bendros paskirties, PC, SP registrai

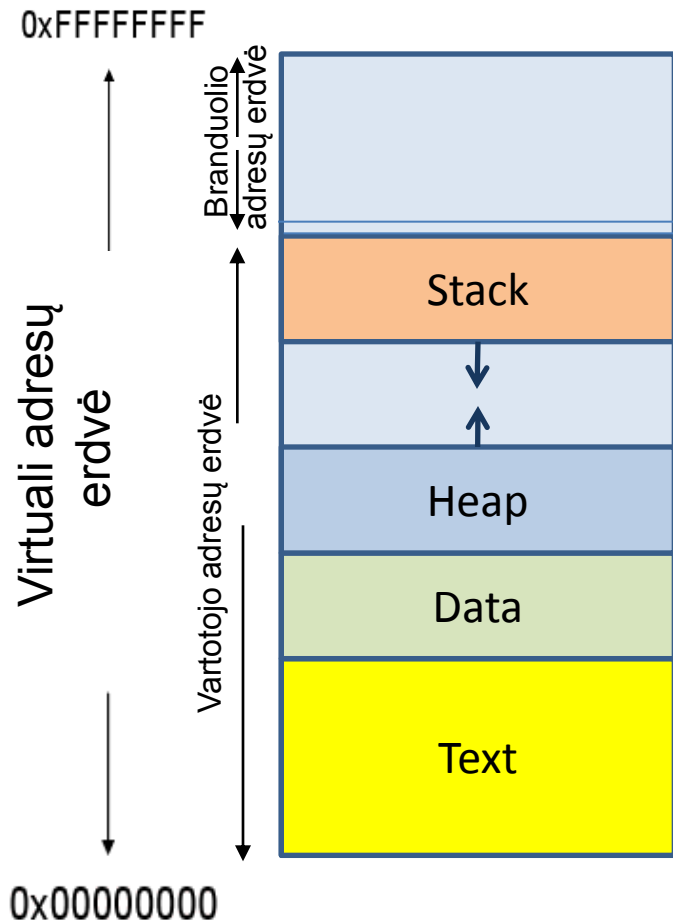
Procesas iš OS perspektyvos

- 3 pagrindiniai komponentai:
 - Proceso adresų erdvė
 - op. atminties dalis, kuri yra pasiekama procesui
 - Skirta įvairių komponentų (programos teksto, kintamųjų, steko, etc.) saugojimui
 - Procesoriaus (CPU) būseną
 - 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)

Procesas atmintyje

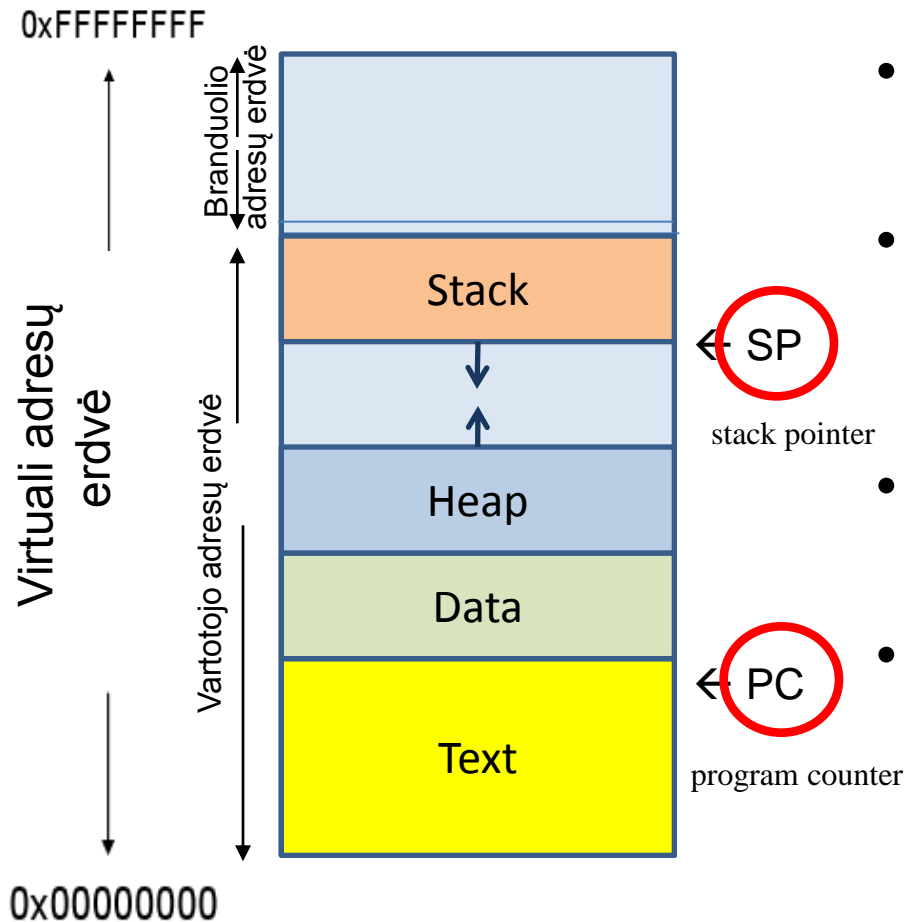


Procesas atmintyje



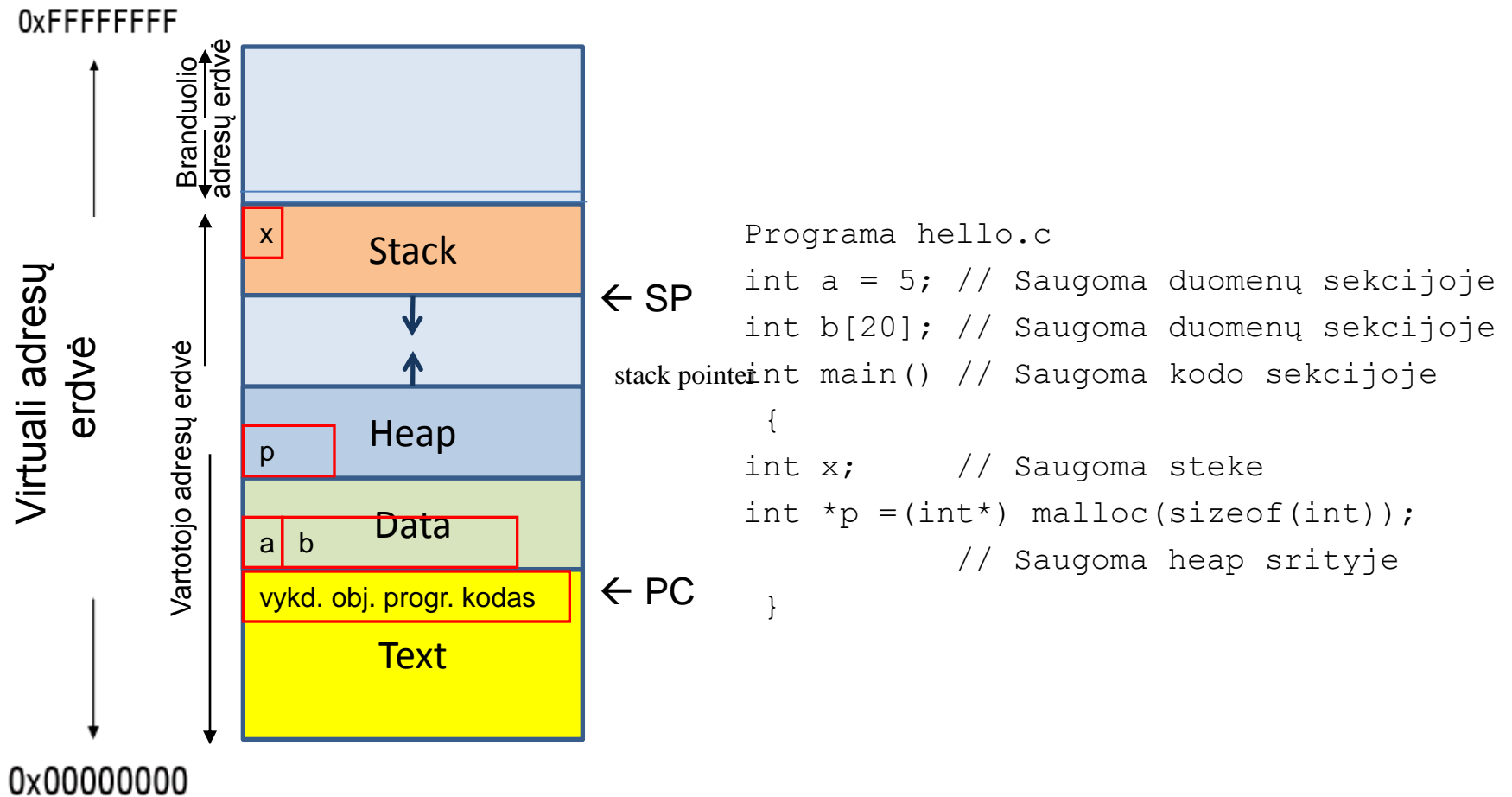
- Steko sritis (stack). Saugomi lokalių funkcijų kintamieji
- Heap sritis. Saugomi dinamiškai kuriami objektai. (C – `malloc()`, C++ – `new`).
- Duomenų sritis (data). Saugomi globalūs kintamieji
- Programos kodo sritis (text). Saugomas objektinis vykdomasis programos kodas

Procesas atmintyje

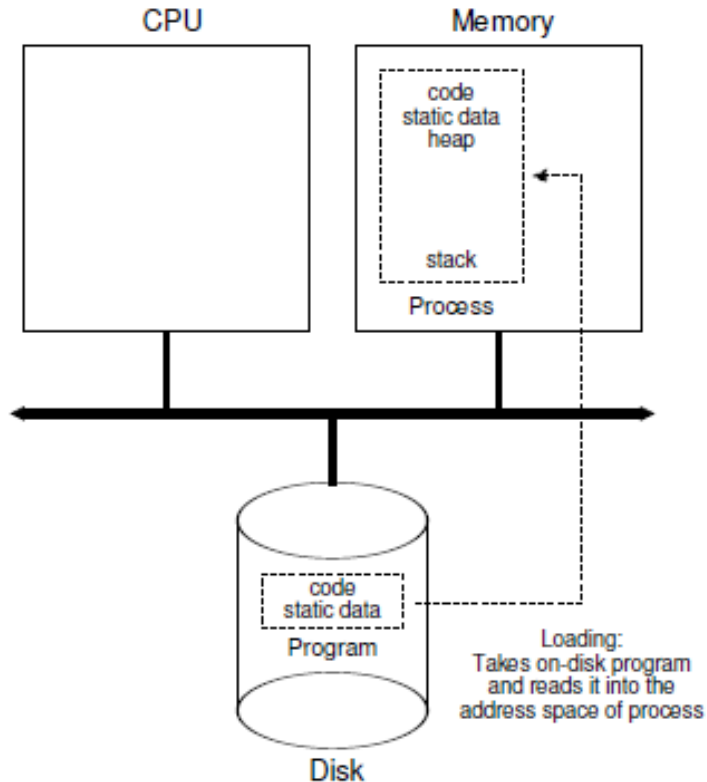


- Steko sritis (stack). Saugomi lokalių funkcijų kintamieji
- Heap sritis. Saugomi dinamiškai kuriami objektai. (C – `malloc()`, C++ – `new`).
- Duomenų sritis (data). Saugomi globalūs kintamieji
- Programos kodo sritis (text). Saugomas objektinis vykdomasis programos kodas

Procesas atmintyje

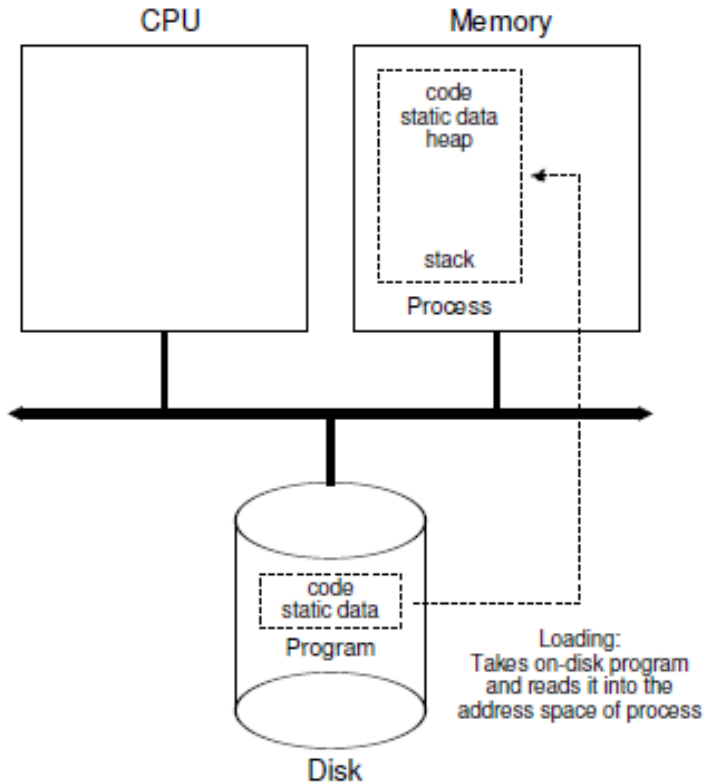


Proceso sukūrimas – detalesnis vaizdas



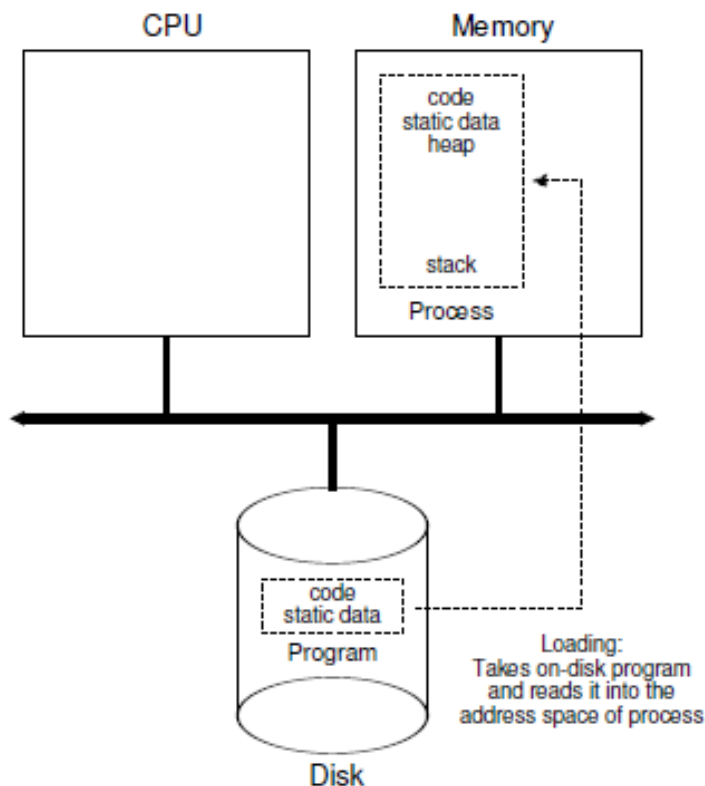
1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas

Proceso sukūrimas – detalesnis vaizdas



1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas
2. Proceso identifikatoriaus (PID) sukūrimas

Proceso sukūrimas – detalesnis vaizdas

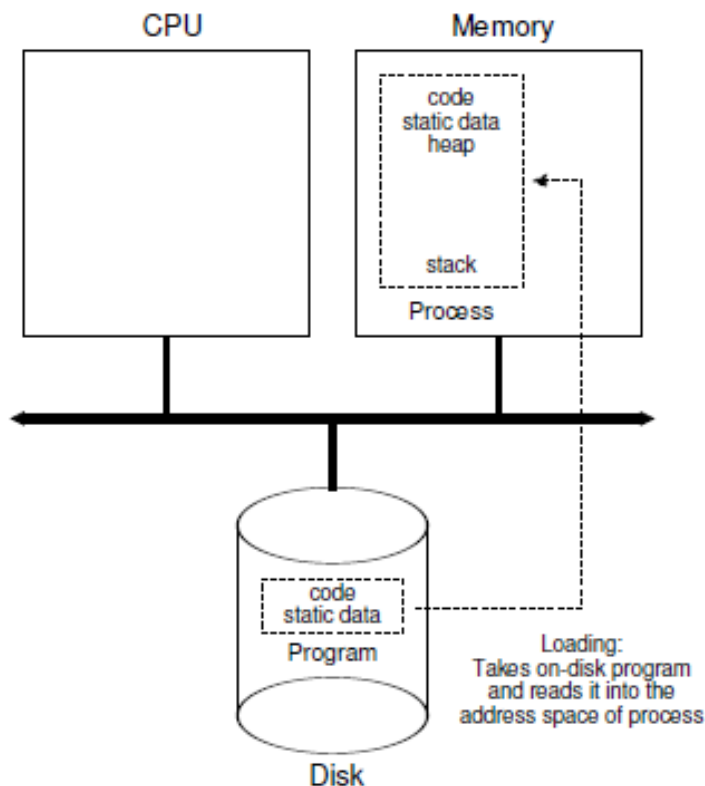


1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas
2. Proceso identifikatoriaus (PID) sukūrimas
3. Vykdytojo programos kodo užkrovimas į atmintį

Programos užkrovimo į atmintį būdai:

- (angl. eagerly)
- (angl. lazily)

Proceso sukūrimas – detalesnis vaizdas

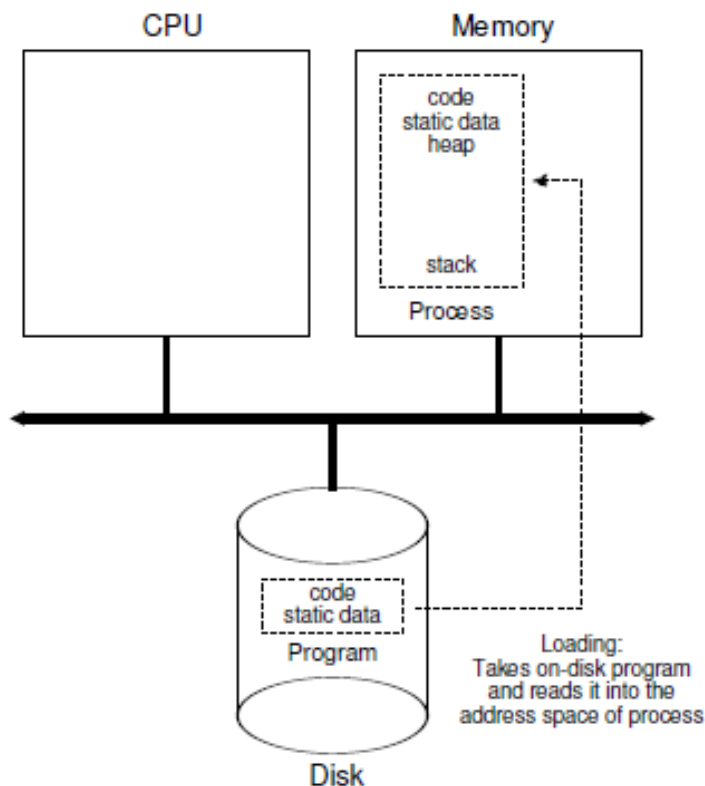


1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas
2. Proceso identifikatoriaus (PID) sukūrimas
3. Vykdytojo programos kodo užkrovimas į atmintį
4. Atminties išskyrimas stekui

Programos užkrovimo į atmintį būdai:

- (angl. eagerly)
- (angl. lazily)

Proceso sukūrimas – detalesnis vaizdas

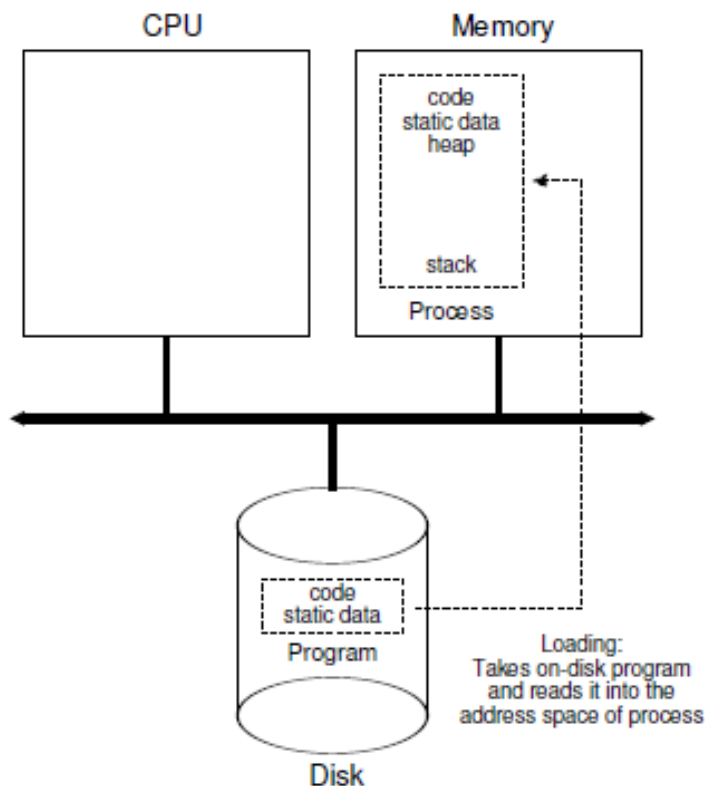


1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas
2. Proceso identifikatoriaus (PID) sukūrimas
3. Vykdytojo programos kodo užkrovimas į atmintį
4. Atminties išskyrimas stekui
5. Atminties išskyrimas heap sričiai

Programos užkrovimo į atmintį būdai:

- (angl. eagerly)
- (angl. lazily)

Proceso sukūrimas – detalesnis vaizdas



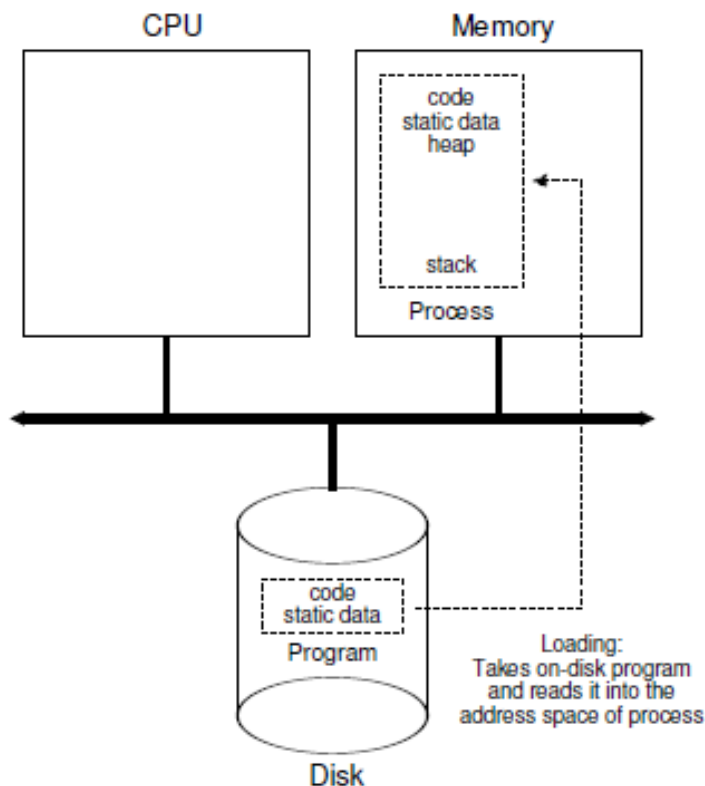
1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas
2. Proceso identifikatoriaus (PID) sukūrimas
3. Vykdytojo programos kodo užkrovimas į atmintį
4. Atminties išskyrimas stekui
5. Atminties išskyrimas heap sričiai
6. Veiksmų susijusių su I/O inicializavimas

Programos užkrovimo į atmintį būdai:

– (angl. eagerly)

– (angl. lazily)

Proceso sukūrimas – detalesnis vaizdas



Programos užkrovimo į atmintį būdai:

– (angl. eagerly)

– (angl. lazily)

1. Duomenų struktūros (PCB) informacijai apie procesą saugoti sukūrimas
2. Proceso identifikatoriaus (PID) sukūrimas
3. Vykdytojo 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)

Proceso būsenos

- Gyvavimo ciklo metu procesas gali įgyti skirtingas būsenas:

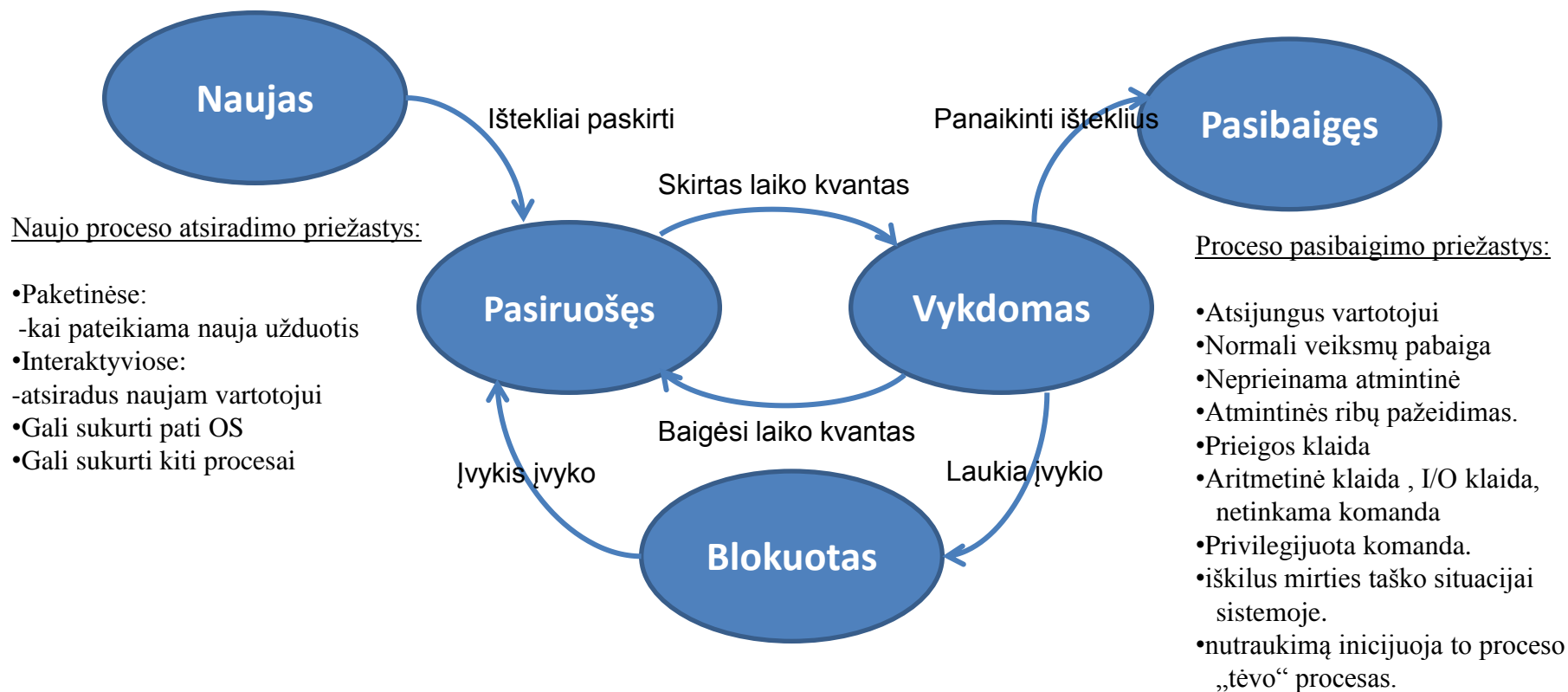
Būsena	Paiškinimas
Naujas (New)	
Pasiruošęs (Ready)	
Vykdomas (Running)	
Blokuotas (Waiting)	
Pasibaigęs (Terminated)	

Proceso būsenos

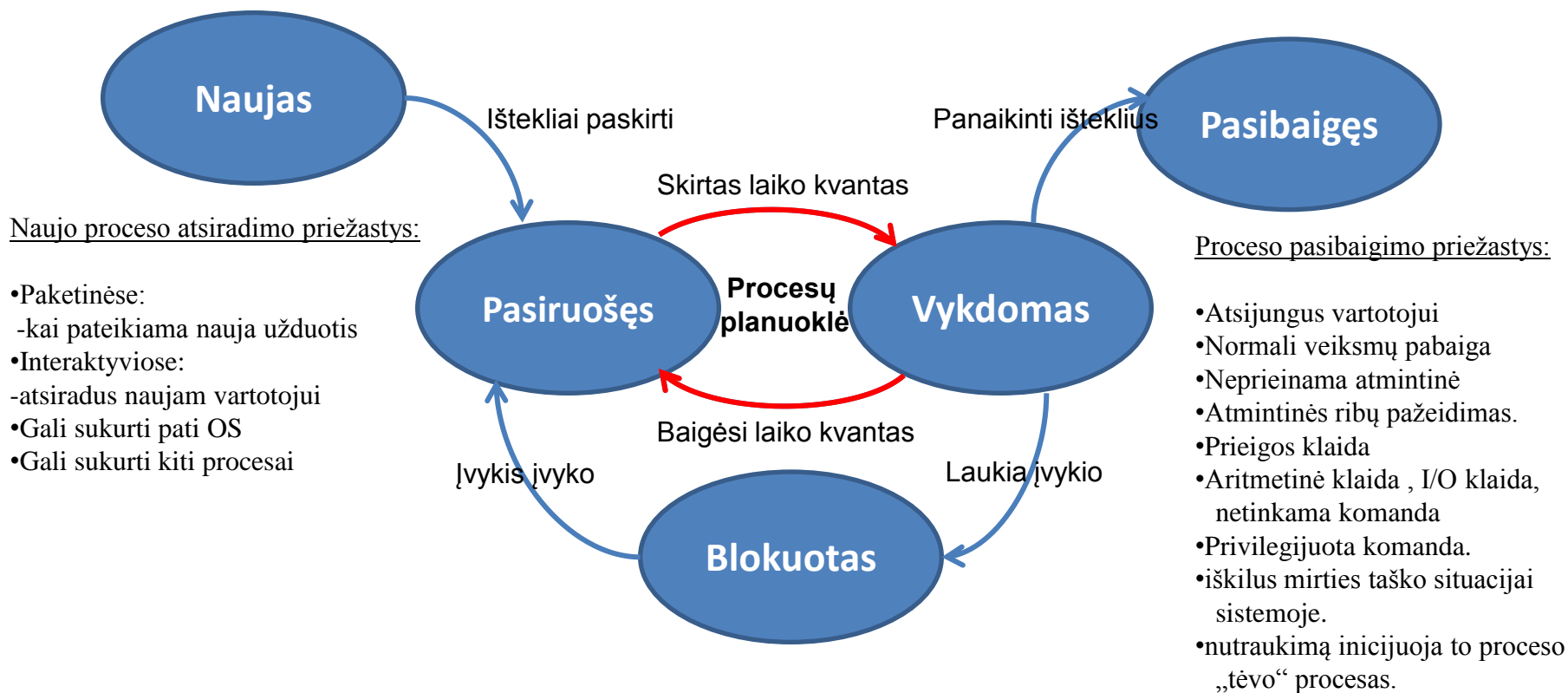
- Gyvavimo ciklo metu procesas gali įgyti skirtingas būsenas:

Būsena	Paiškinimas
Naujas (New)	ką tik sukurtas
Pasiruošęs (Ready)	laukia eilėje prie CPU resursų
Vykdomas (Running)	CPU ištekliai procesui priskirti
Blokuotas (Waiting)	laukia pasirodančio kokio nors įvykio
Pasibaigęs (Terminated)	baigė darbus

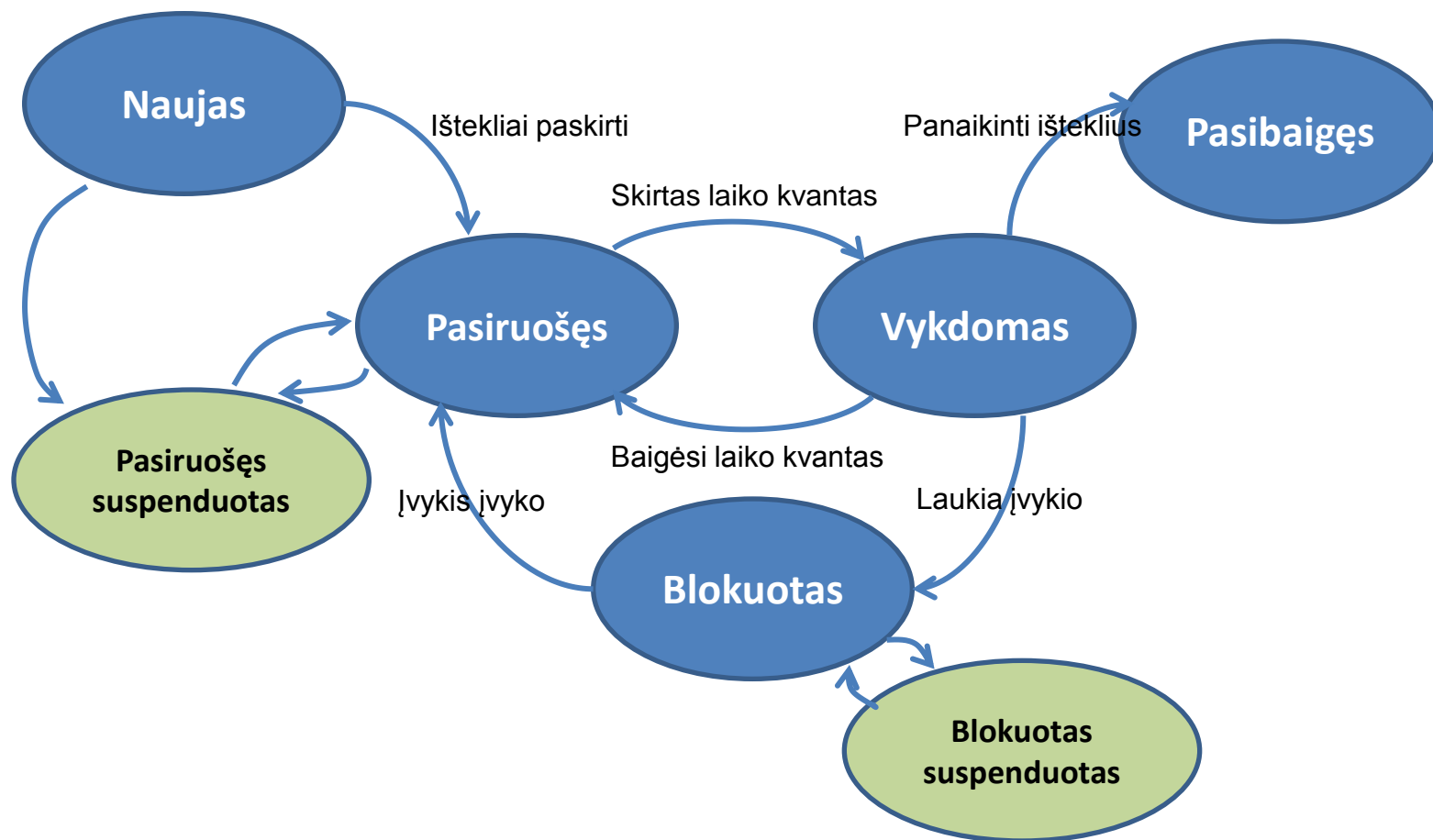
Proceso būsenų diagrama (5 būsenų)



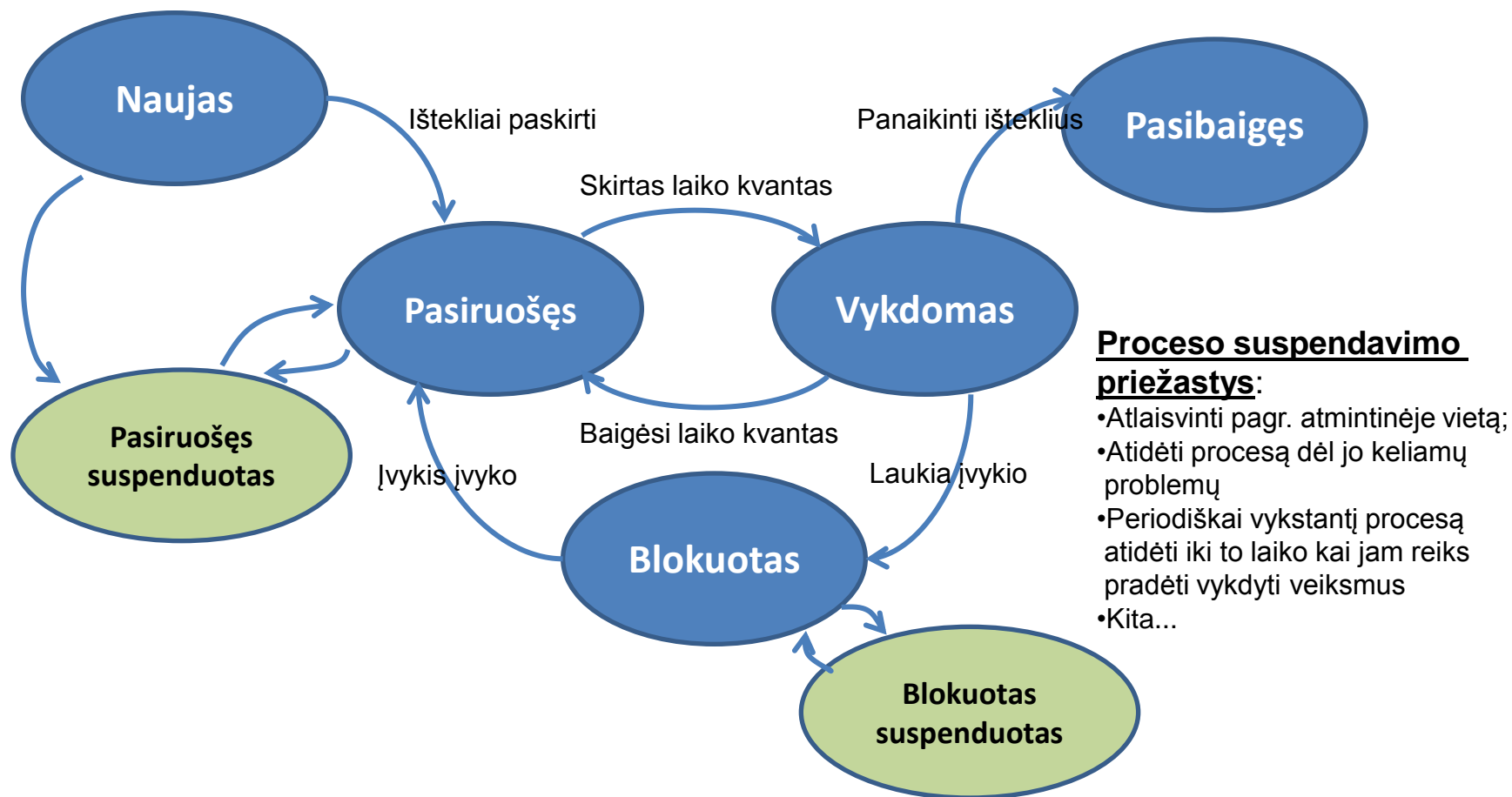
Proceso būsenų diagrama (5 būsenų)



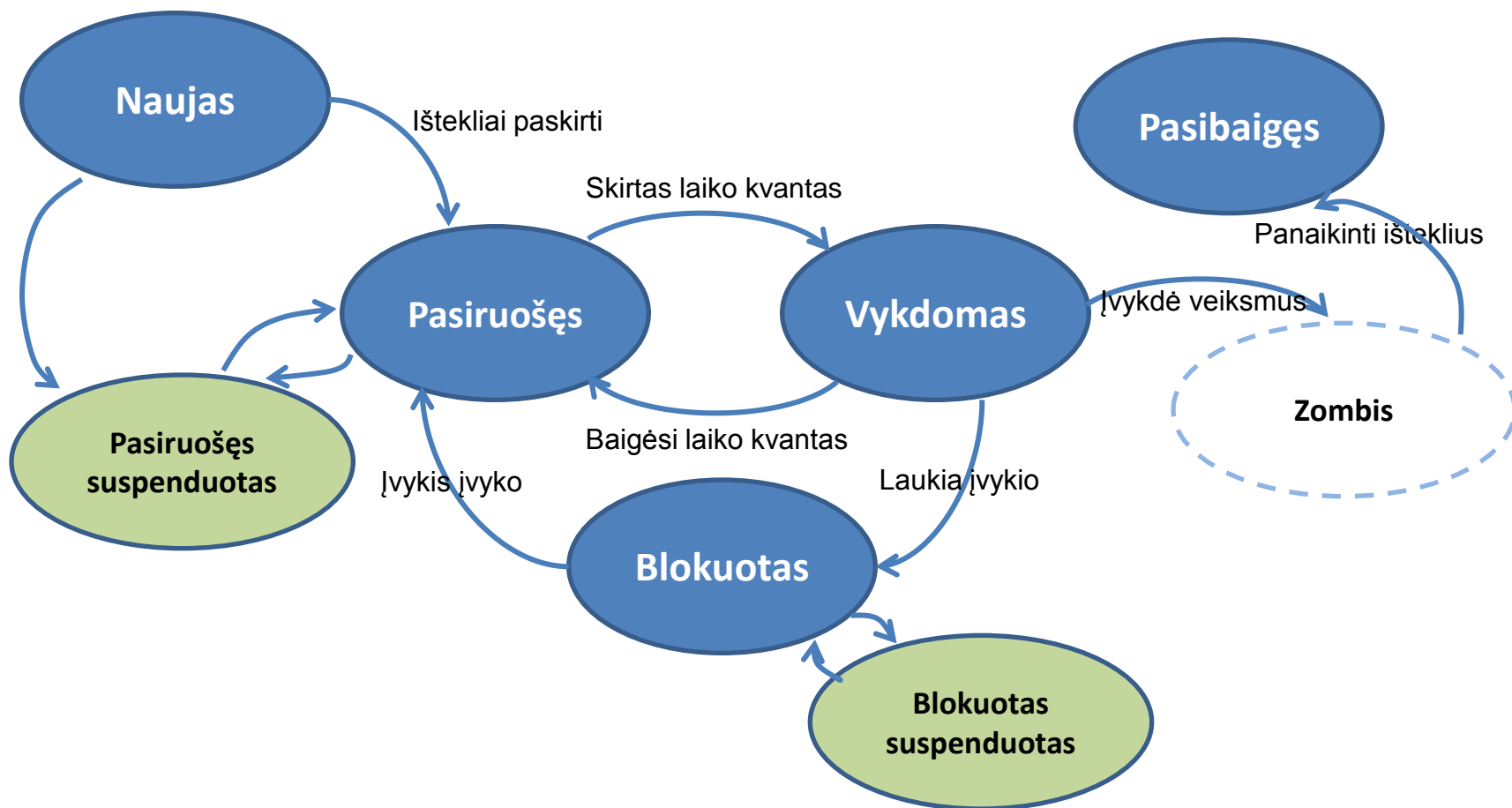
Proceso būsenų diagrama (7 būsenų)



Proceso būsenų diagrama (7 būsenų)



Proceso būsenų diagrama (7 būsenų)



Proceso būsenos. Pavyzdžiai

Time	Process ₀	Process ₁	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	
4	Running	Ready	Process ₀ now done
5	–	Running	
6	–	Running	
7	–	Running	
8	–	Running	Process ₁ now done

a)

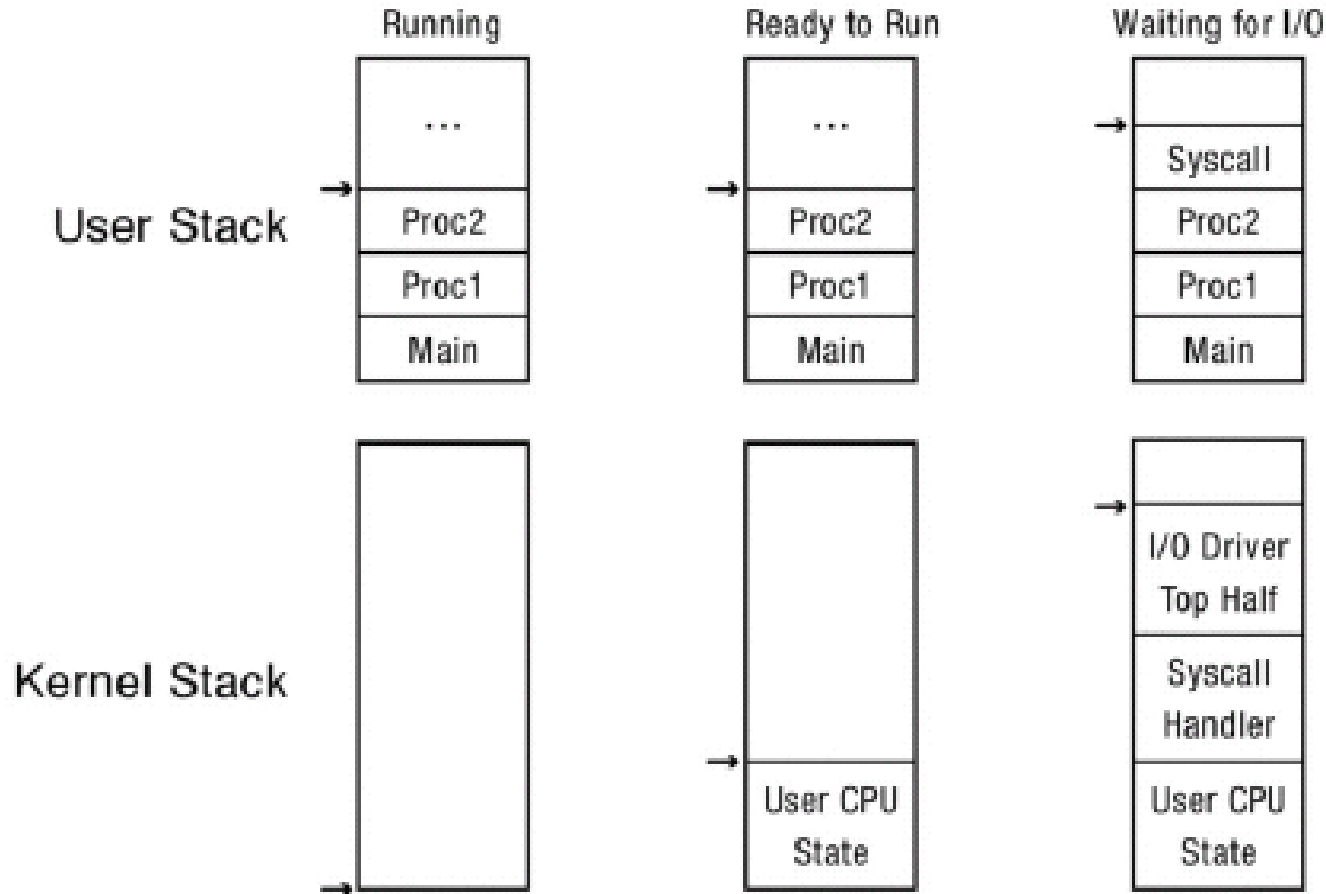
Tik skaičiavimo tipo procesai

Time	Process ₀	Process ₁	Notes
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	Process ₀ initiates I/O
4	Blocked	Running	Process ₀ is blocked, so Process ₁ runs
5	Blocked	Running	
6	Blocked	Running	
7	Ready	Running	I/O done
8	Ready	Running	Process ₁ now done
9	Running	–	
10	Running	–	Process ₀ now done

b)

Skaičiavimo, I/O tipo procesai

Branduolio ir vartotojo stekai skirtingos būsenos procesams

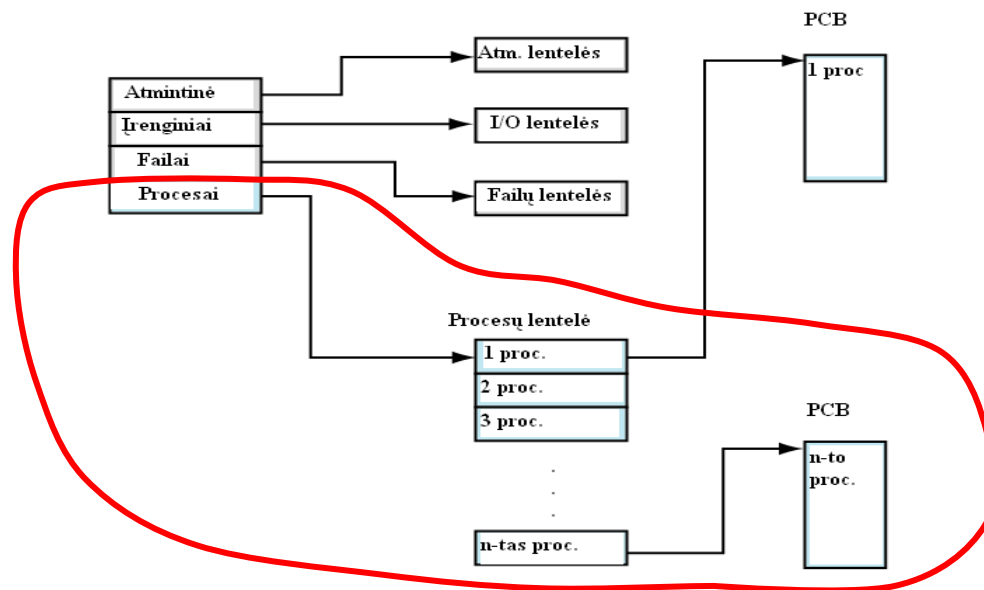


Procesų valdymui naudojamos duomenų struktūros

- Procesų lentelė
- Proceso kontrolės blokas
- Procesų eilės

Procesų lentelė

- OS info apie procesus saugo procesų lentelėje (angl. Process Table (PT):
 - PID – identifikatorius (arba rodyklė) į PT įrašą;
 - PT įrašas = Proceso kontrolės blokas (PCB).



Proceso kontrolės blokas (PCB)

Rodyklė į sekantį PCB	proceso būvis
Rodyklė į prieš tai esantį PCB	
Proceso ID (PID)	
programos skaitliukas	
registrai	
Atmintinės struktūra	
Atvertų failų lentelė	
Ir t.t.	

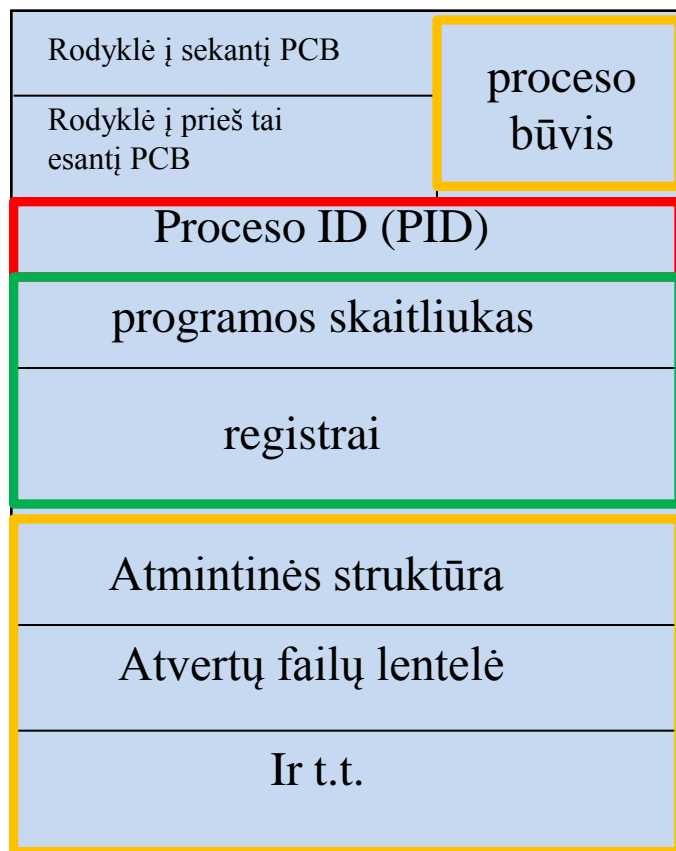
- Proceso identifikaciniai duomenys (proceso ID, tėvo proceso ID, vartotojo, sukūrusio procesą ID, grupė ir t.t)

Proceso kontrolės blokas (PCB)

Rodyklė į sekantį PCB	proceso būvis
Rodyklė į prieš tai esantį PCB	
Proceso ID (PID)	
programos skaitliukas	
registrai	
Atmintinės struktūra	
Atvertų failų lentelė	
Ir t.t.	

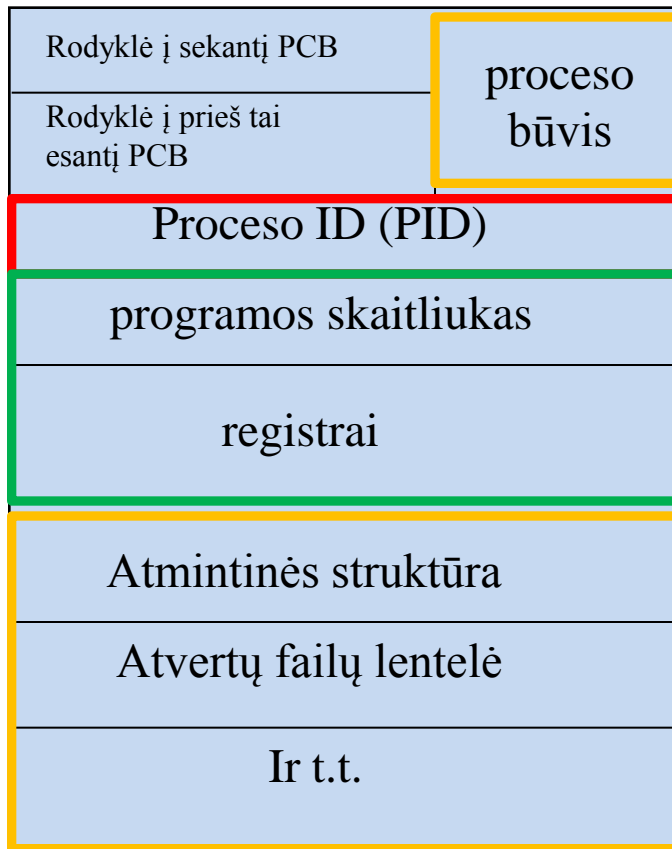
- Proceso identifikaciniai duomenys (proceso ID, tėvo proceso ID, vartotojo, sukūrusio procesą ID, grupė ir t.t)
- Procesoriaus būsenos duomenys (PC, SP, PSW, bendros paskirties registrų turiniai ir t.t.)

Proceso kontrolės blokas (PCB)



- Proceso identifikaciniai duomenys (proceso ID, tėvo proceso ID, vartotojo, sukūrusio procesą ID, grupė ir t.t.)
- Procesoriaus būsenos duomenys (PC, SP, PSW, bendros paskirties registrų turiniai ir t.t.)
- Proceso valdymui skirti duomenys (proceso būseną, vykdymo prioritetą, privilegijas, kita statistinė info)

Proceso kontrolės blokas (PCB)



PROCESO KONTEKSTAS

- Proceso identifikaciniai duomenys (proceso ID, tėvo proceso ID, vartotojo, sukūrusio procesą ID, grupė ir t.t.)
- Procesoriaus būsenos duomenys (PC, SP, PSW, bendros paskirties registrų turiniai ir t.t.)
- Proceso valdymui skirti duomenys (proceso būseną, vykdymo prioritetą, privilegijas, kita statistinė info)

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  

```

Visa struktūra: include/linux/sched.h

```
};
```

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  

```

Visa struktūra: include/linux/sched.h

```
};
```

task_struct fragmentas (Linux OS)

```
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;  
  
    pid_t pid;  
    pid_t tgid;  
  
    struct task_struct *real_parent;  
  
    char comm[TASK_COMM_LEN];  
  
    struct thread_struct thread;  
  
    struct files_struct *files;  
  
    ...  
};
```

Visa struktūra: include/linux/sched.h

Xv6 proceso struktūra

```
// the registers xv6 will save and restore
// to stop and subsequently restart a process
struct context {
    int eip;
    int esp;
    int ebx;
    int ecx;
    int edx;
    int esi;
    int edi;
    int ebp;
};

// the different states a process can be in
enum proc_state { UNUSED, EMBRYO, SLEEPING,
                  RUNNABLE, RUNNING, ZOMBIE };

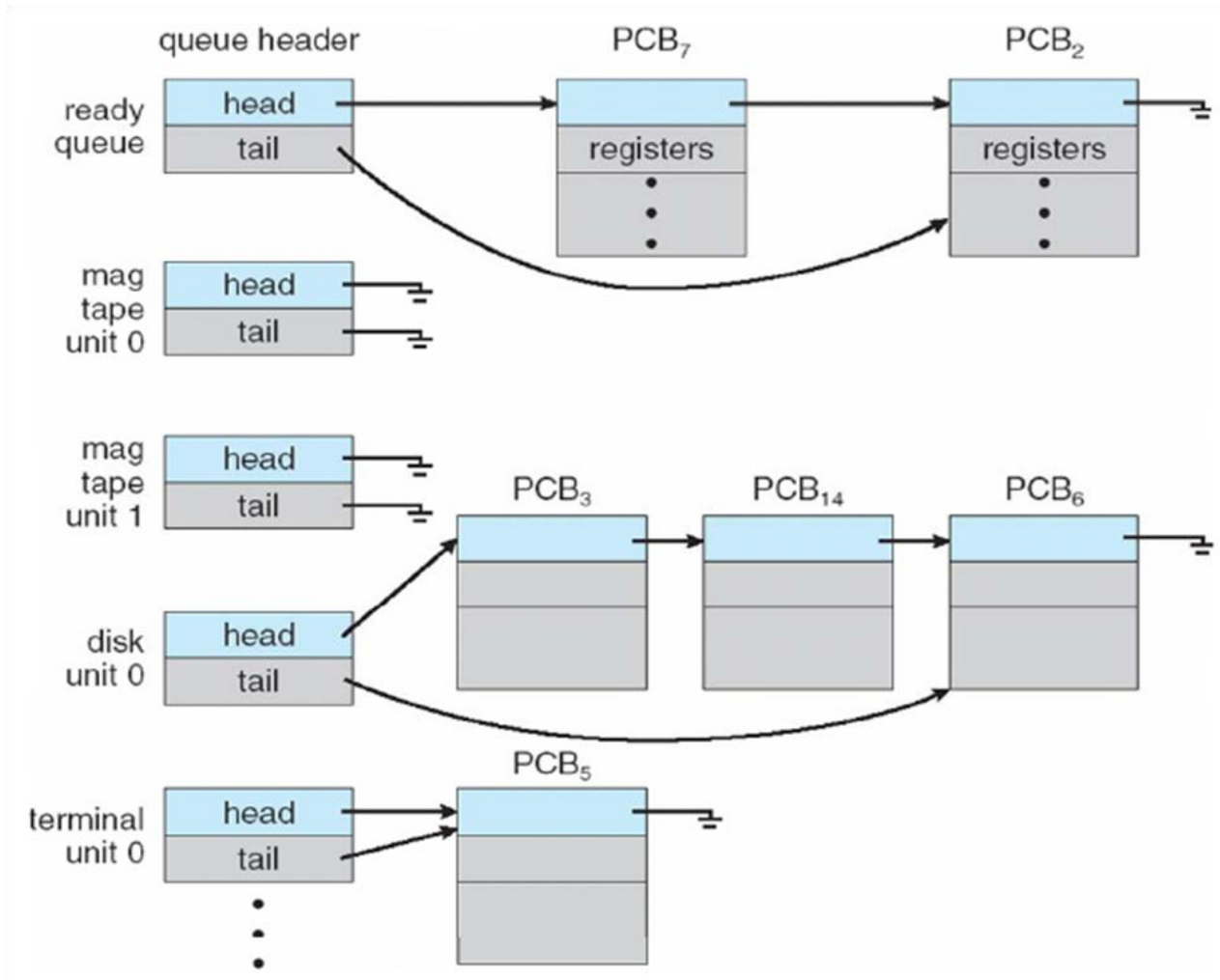
// the information xv6 tracks about each process
// including its register context and state
struct proc {
    char *mem;                // Start of process memory
    uint sz;                  // Size of process memory
    char *kstack;             // Bottom of kernel stack
                                // for this process
    enum proc_state state;    // Process state
    int pid;                  // Process ID
    struct proc *parent;      // Parent process
    void *chan;               // If non-zero, sleeping on chan
    int killed;               // If non-zero, have been killed
    struct file *ofile[NOFILE]; // Open files
    struct inode *cwd;         // Current directory
    struct context context;    // Switch here to run process
    struct trapframe *tf;     // Trap frame for the
                                // current interrupt
};
```

Procesų eilės – pagal poreikius resursams

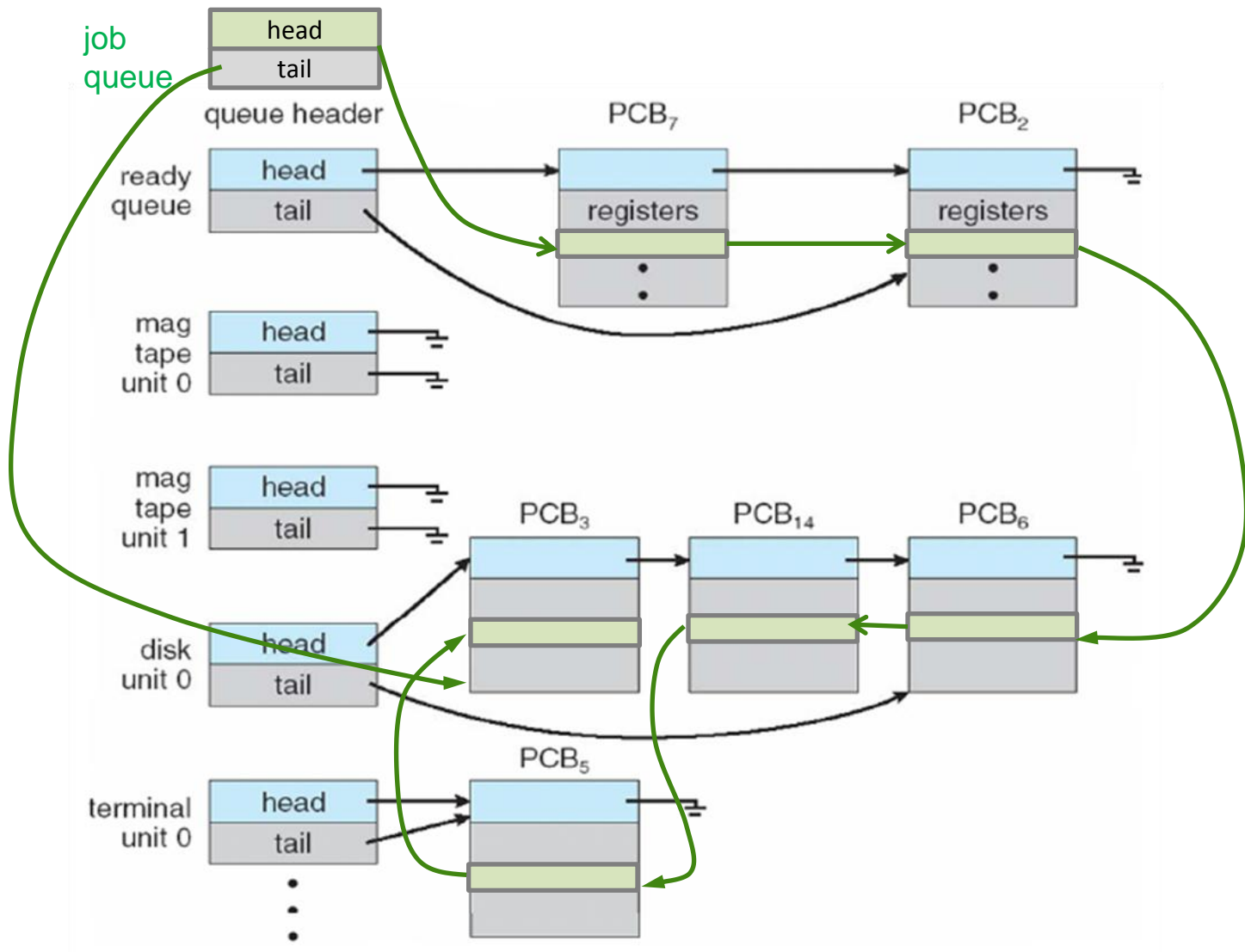
- Bendra procesų eilė (angl. job queue) – visų sistemos procesų eilė
- Pasiruošusių procesų eilė (angl. ready queue) – atmintyje esančių procesų, laukiančių vykdymo procesoriuje eilė
- Įrenginių eilės – procesų, laukiančių prieigos prie tam tikro įrenginio resursų, eilė

Procesai tarp skirtingo tipo eilių gali migruoti

Pasiruošusių procesų ir skirtingų įrenginių eilės



Pasiruošusių procesų ir skirtingų įrenginių eilės



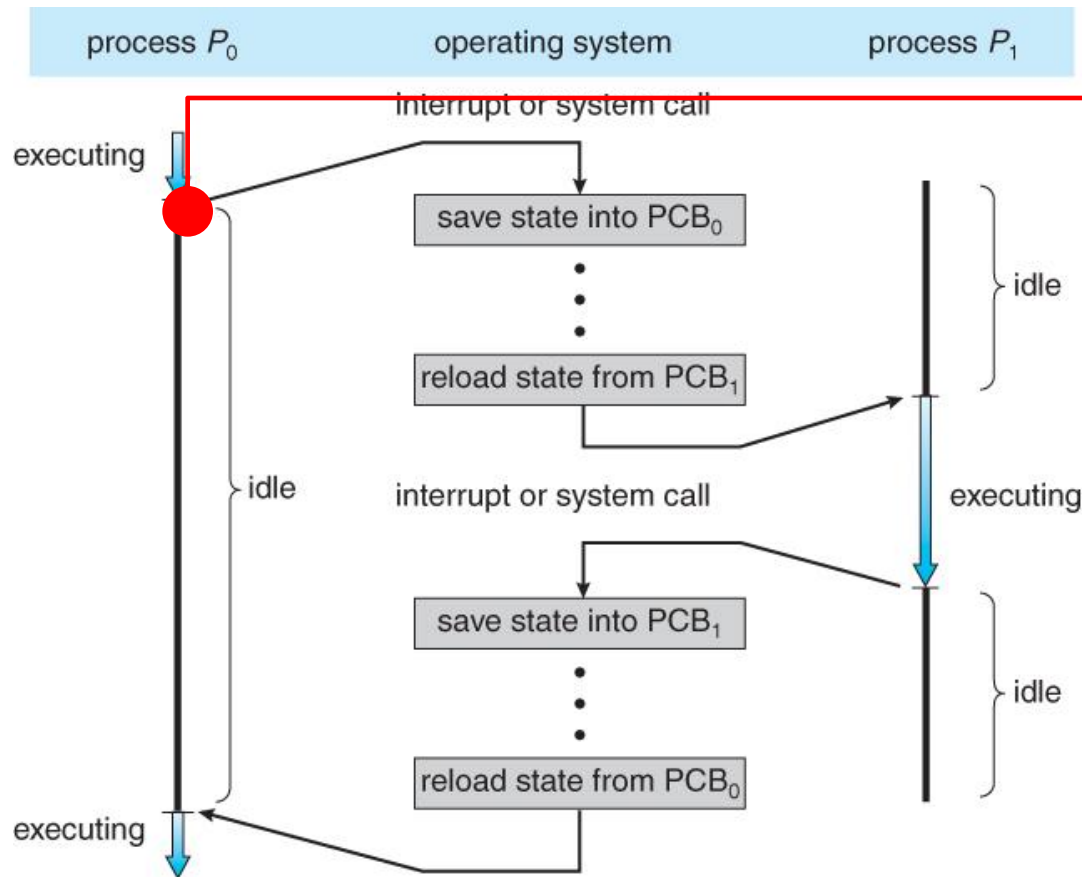
Persijungimas nuo vieno proceso prie kito

- Procesas vykdomas – kaip OS atgauti kontrolę?

Persijungimas nuo vieno proceso prie kito

- Procesas vykdomas – kaip OS atgauti kontrolę?
 - Kooperatyvus būdas (sisteminio kreipinio metu, įvykus klaidai. `yield()` sisteminio kreipinio naudojimas)
 - Nekooperatyvus būdas (laikrodžio pertraukimo mechanizmo įvedimas)

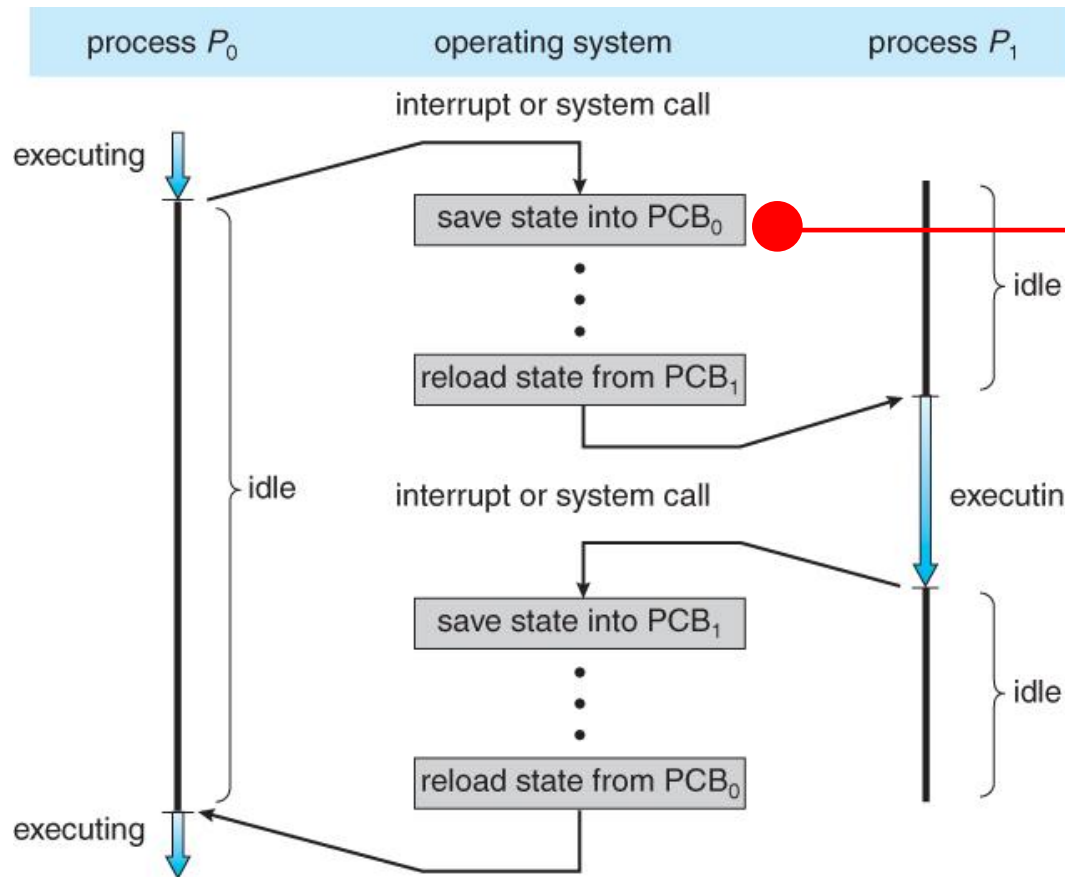
Perėjimo nuo vieno proceso prie kito procedūra



Įvykiai inicijuojantys proceso perjungimą:

- CPU laiko kvanto pabaiga;
- I/O pertraukimas;
- Klaidos (susijusios su kreipiniais į pagr. atmintį, procesų vykdymu ir pan.)

Perėjimo nuo vieno proceso prie kito procedūra

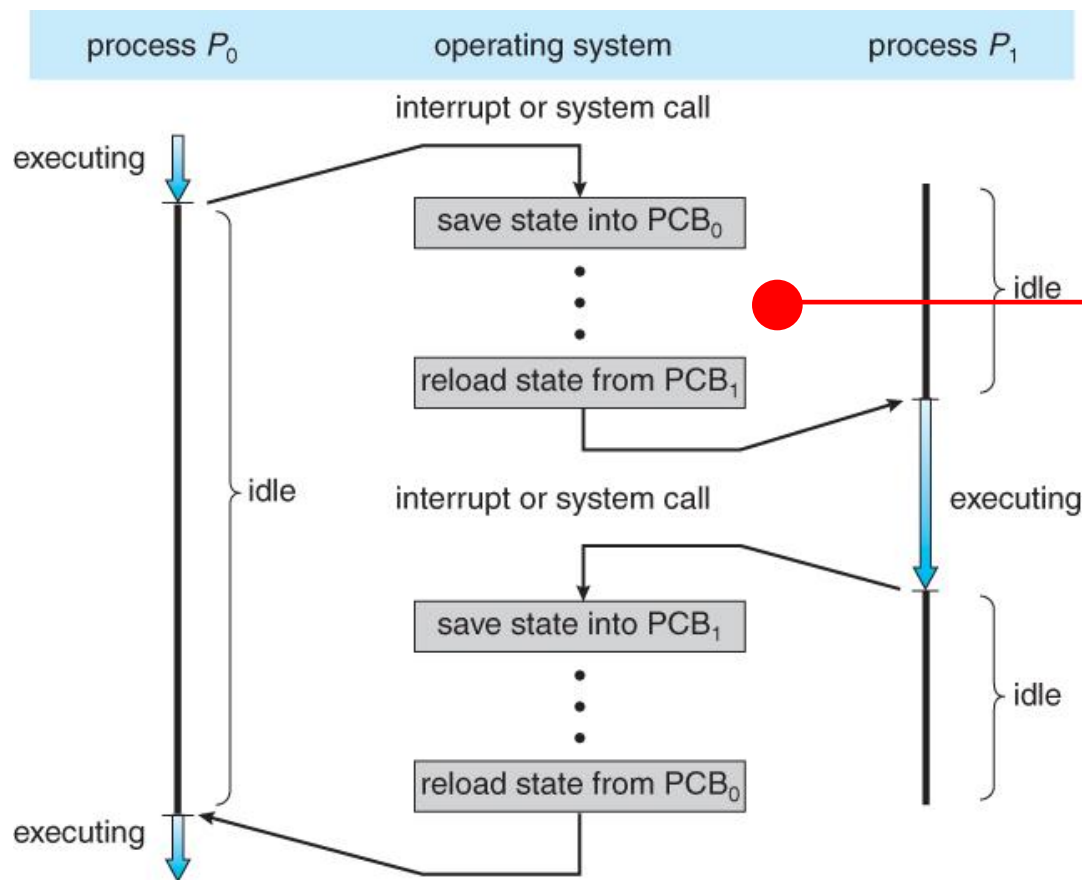


Įvykiai inicijuojantys proceso perjungimą:

- CPU laiko kvanto pabaiga;
- I/O pertraukimas;
- Klaidos (susijusios su kreipiniais į pagr. atmintį, procesų vykdymu ir pan.)

prieš tai vykdyto proceso kontekstas yra išsaugomas to proceso kontrolės bloke (PCB). Keičiamas jo būvis.

Perėjimo nuo vieno proceso prie kito procedūra



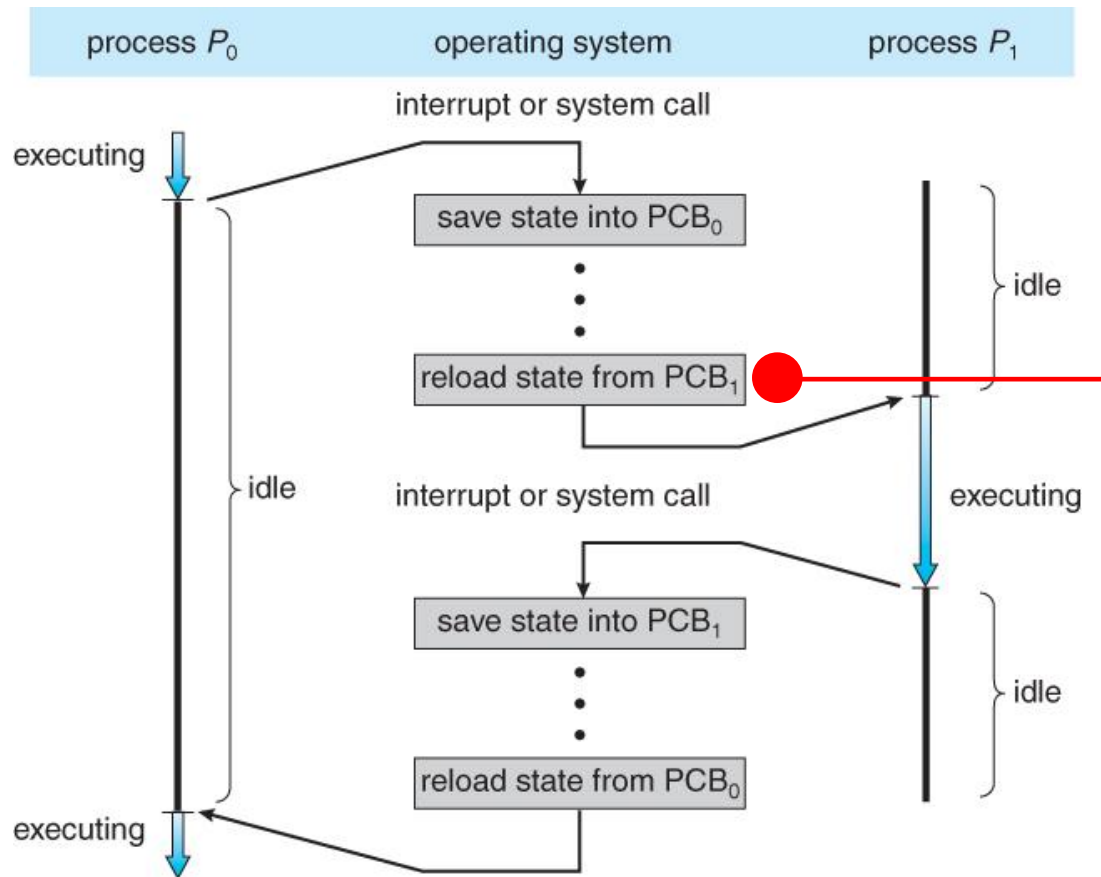
Įvykiai inicijuojantys proceso perjungimą:

- CPU laiko kvanto pabaiga;
- I/O pertraukimas;
- Klaidos (susijusios su kreipiniais į pagr. atmintį, procesų vykdymu ir pan.)

prieš tai vykdyto proceso kontekstas yra išsaugomas to proceso kontrolės bloke (PCB). Keičiamas jo būvis.

Vykdymui parenkamas naujas procesas

Perėjimo nuo vieno proceso prie kito procedūra



Įvykiai inicijuojantys proceso perjungimą:

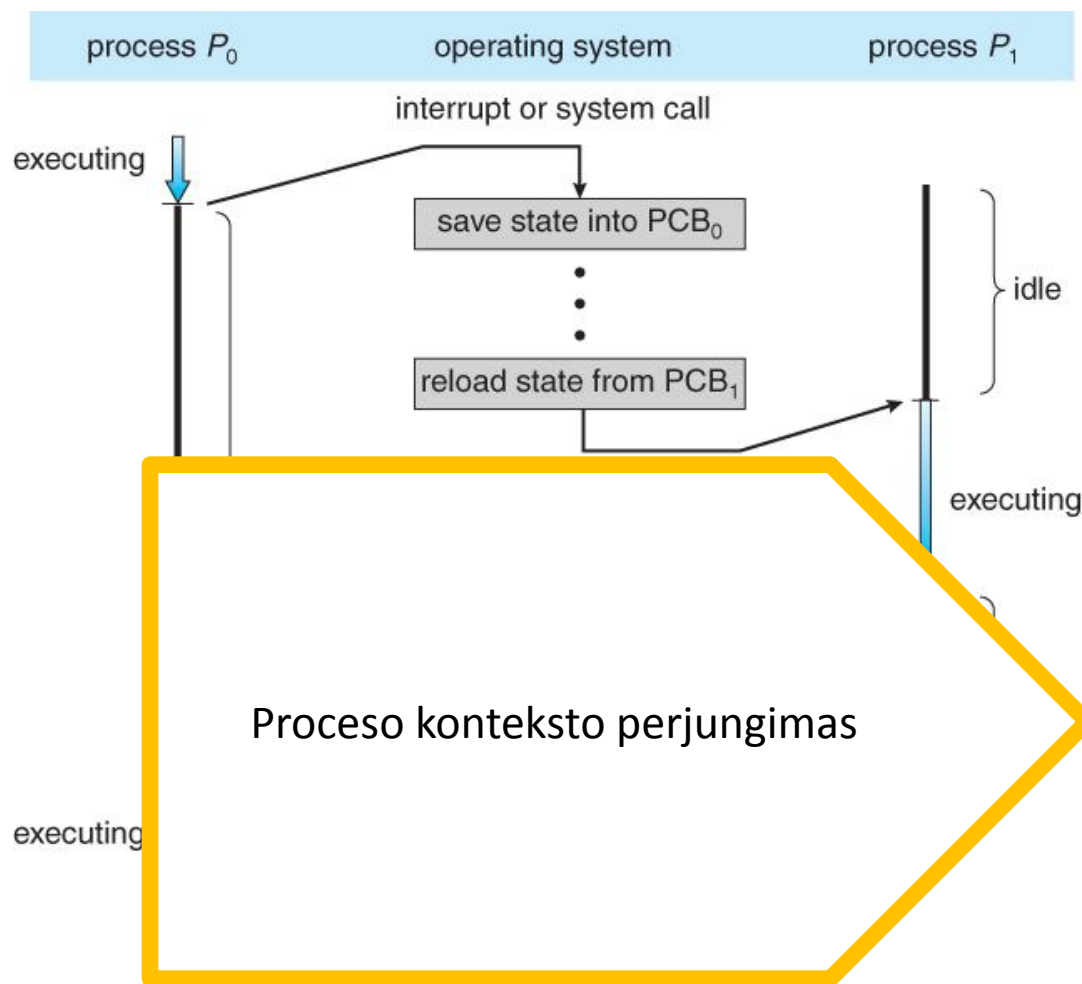
- CPU laiko kvanto pabaiga;
- I/O pertraukimas;
- Klaidos (susijusios su kreipiniais į pagr. atmintį, procesų vykdymu ir pan.)

prieš tai vykdyto proceso kontekstas yra išsaugomas to proceso kontrolės bloke (PCB). Keičiamas jo būvis.

Vykdymui parenkamas naujas procesas

įkraunamas naujo proceso kontekstas (iš jo PCB bloko). Keičiamas jo būvis į vykdomą

Perėjimo nuo vieno proceso prie kito procedūra



Įvykiai inicijuojantys proceso perjungimą:

- CPU laiko kvanto pabaiga;
- I/O pertraukimas;
- Klaidos (susijusios su kreipiniais į pagr. atmintį, procesų vykdymu ir pan.)

prieš tai vykdyto proceso kontekstas yra išsaugomas to proceso kontrolės bloke (PCB). Keičiamas jo būvis.

Vykdyti parenkamas naujas procesas

įkraunamas naujo proceso kontekstas (iš jo PCB bloko). Keičiamas jo būvis į vykdomą

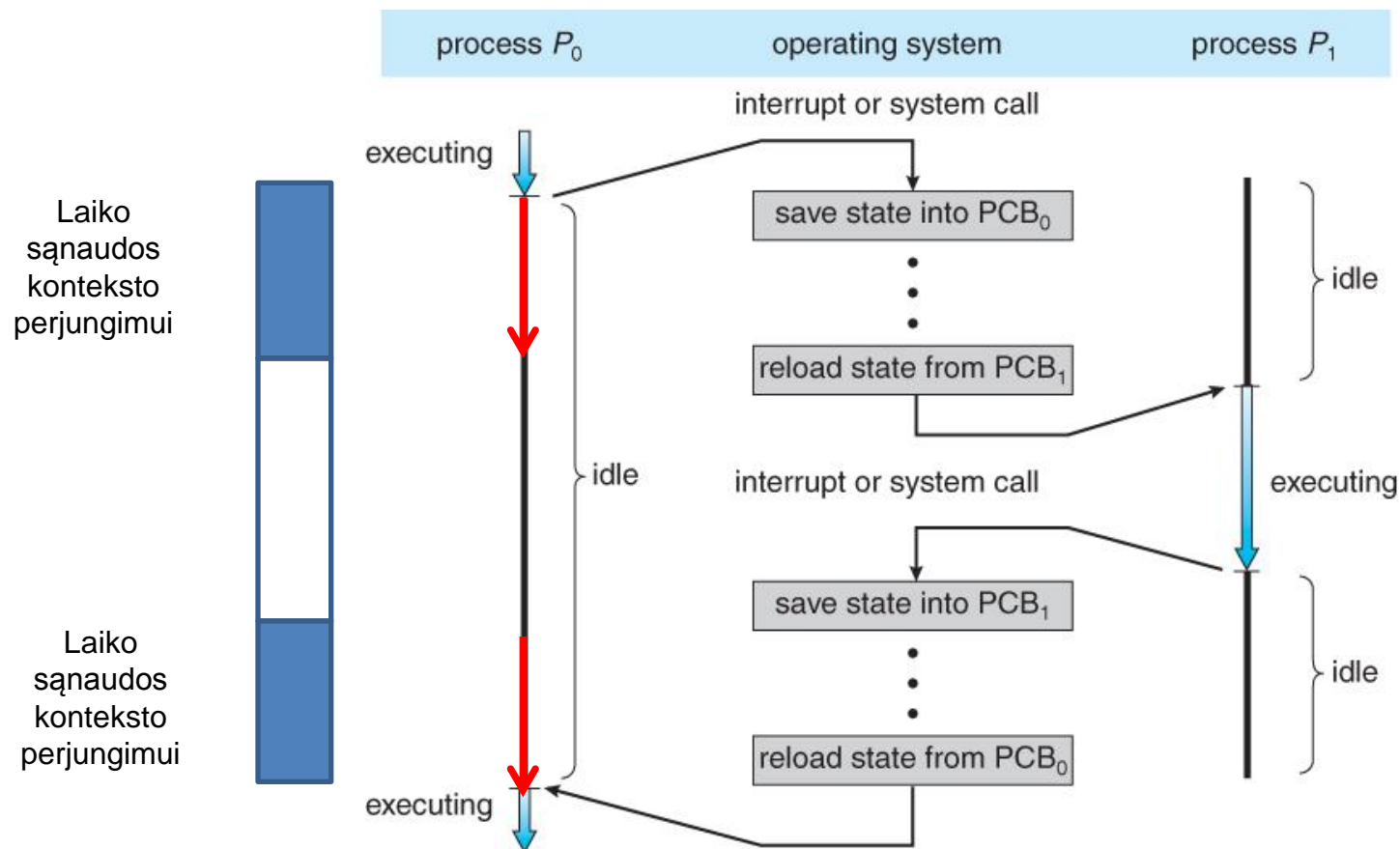
Perėjimo nuo vieno proceso prie kito procedūra

- Proceso konteksto perjungimo metu išsaugoma:
 - Programos skaitiklis (PC)
 - Bazinio ir ribinio registrų reikšmės
 - Perjungimo metu naudoti registrai ir jų turiniai
 - Pasikeitusi proceso būseną
 - I/O būseną
 - Su proceso vykdymo planavimu susijusi info
 - Kita statistinė info.

Proceso konteksto perjungimas – detalesnis vaizdas

OS @ boot (kernel mode)	Hardware	
initialize trap table	remember addresses of... syscall handler timer handler	
start interrupt timer	start timer interrupt CPU in X ms	
OS @ run (kernel mode)	Hardware	Program (user mode)
		Process A
		...
	timer interrupt save regs(A) to k-stack(A) move to kernel mode jump to trap handler	
Handle the trap Call switch() routine save regs(A) to proc-struct(A) restore regs(B) from proc-struct(B) switch to k-stack(B) return-from-trap (into B)		
	restore regs(B) from k-stack(B) move to user mode jump to B's PC	
		Process B
		...

Perėjimo nuo vieno proceso prie kito procedūra



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

Klausimai?

- Paskaitos medžiaga...?
- PR1...?
- Papildomos veiklos (Bonus taškai)...?