Procesorski sistemi v telekomunikacijah
Operacijski sistem, večopravilnost

Operacijski sistem (OS)

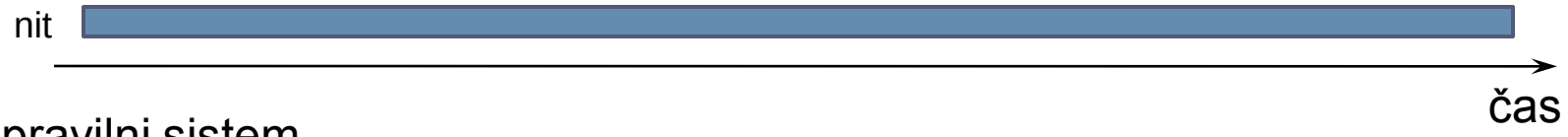
- **Operacijski sistem** - zbirka programov in podatkov, ki upravljajo s strojno opremo in nudijo storitve ostalim programom.
- **API** (Application programming Interface) - vmesnik preko katerega programi dostopajo do storitev (bodisi storitev OS, ali kakih drugih).
- Prošnja za izvedbo storitve OS – **sistemski klic**.
- Primer: Želimo zapisati podatke na trdi disk.
  Program ne piše neposredno v registre krmilnika trdega diska.
  Namesto tega pripravi podatke in izvede sistemski klic za zapis podatkov na disk.
  Vso komunikacijo s krmilnikom trdega diska opravi OS.
  Prednost: ob zamenjavi krmilnika z drugačnim je potrebno le spremeniti del OS, ki upravlja z diskom (gonilnik), ne pa tudi vseh programov.



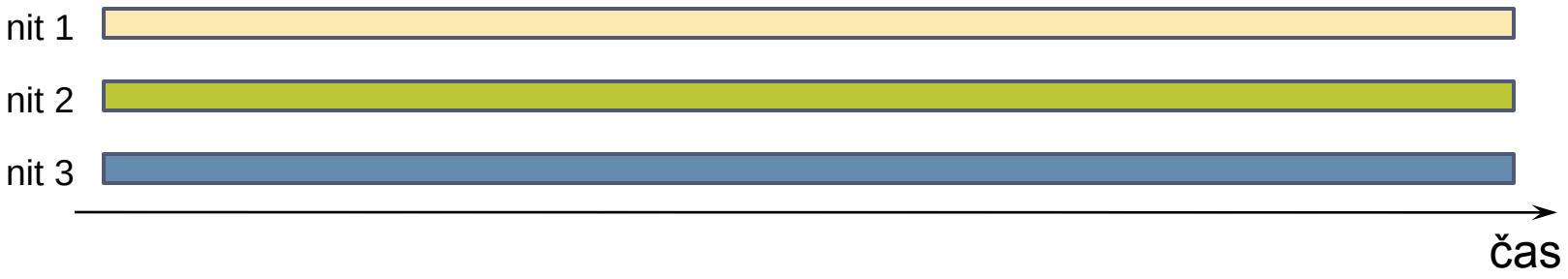

Večopravilnost (Multitasking)
Nit (Thread)

- Možnost sočasnega izvajanja večih programov na enem procesorju.
- Sočasnost je le navidezna – naenkrat se izvaja le en program.
- Program v večopravilnem sistemu ... nit.
- (Navidez) sočasno izvajanje omogoča operacijski sistem.
- SMT mikroprocesorji imajo strojno podporo za sočasno izvajanje omejenega števila programov na enem procesorju (jedru).

Enoopravilni sistem

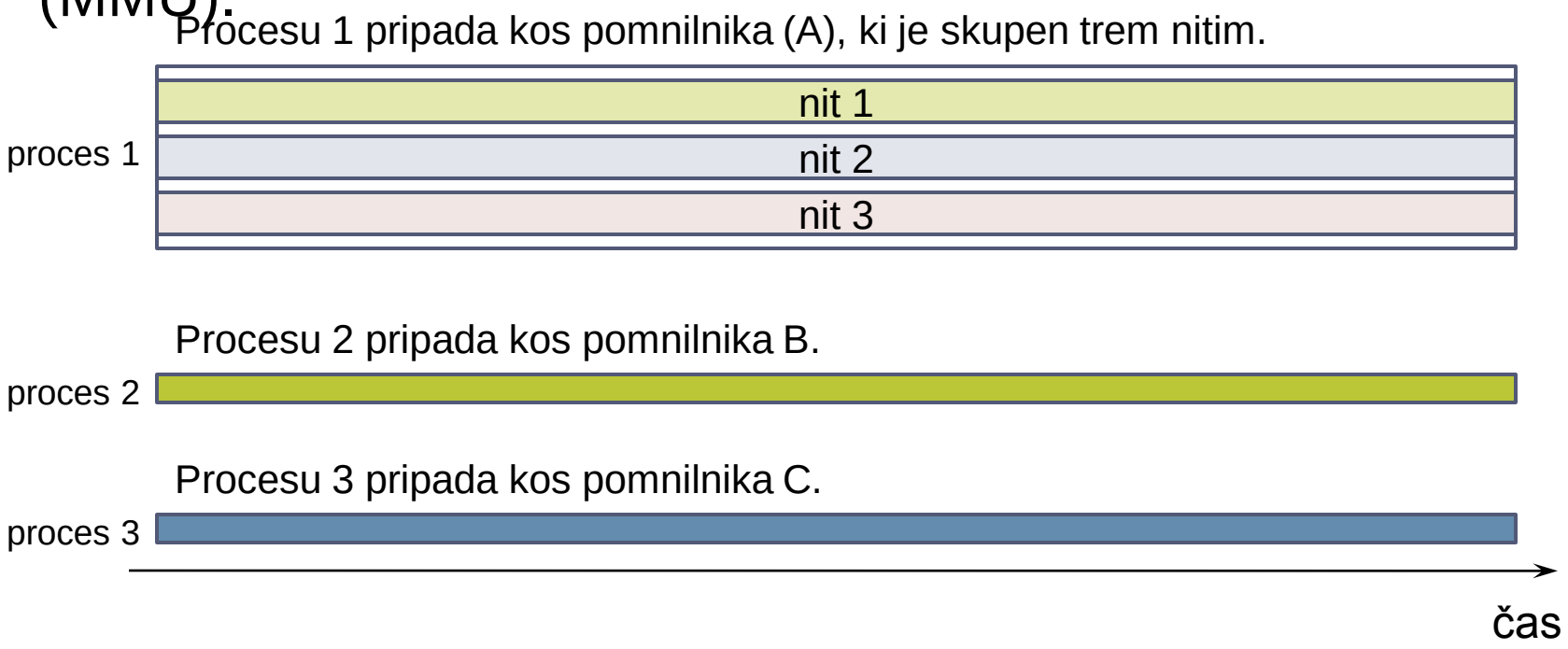


Večopravilni sistem



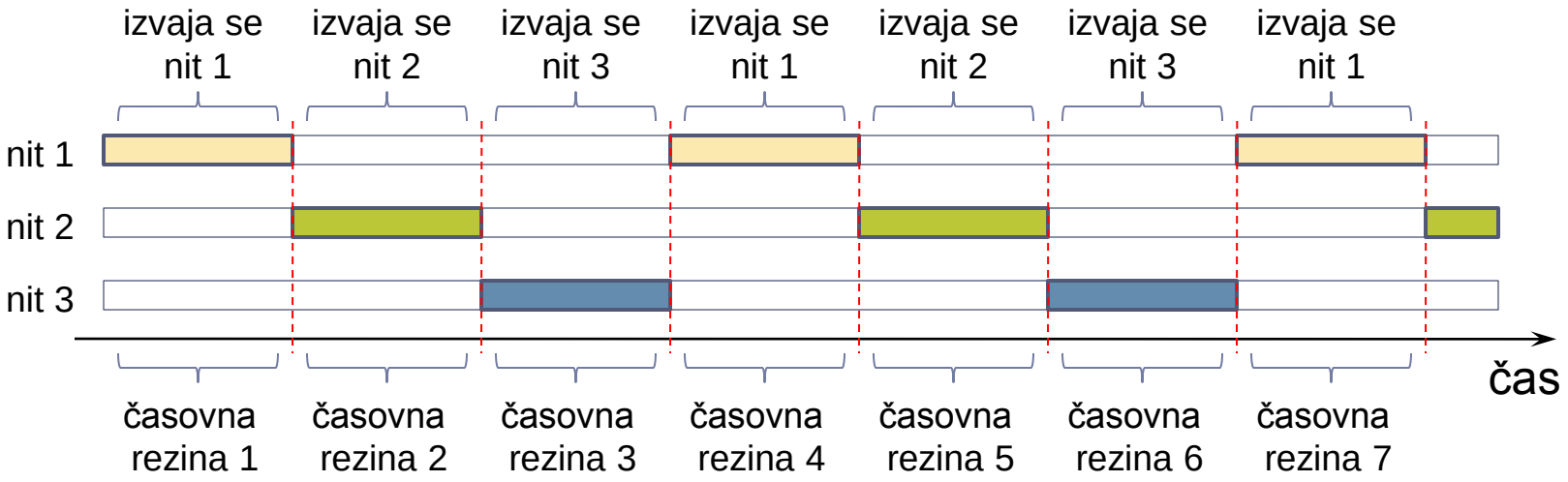
Večnitni (Multithreaded) procesi

- Niti, ki si delijo skupen pomnilnik pripadajo enemu procesu.
- Niti nekega procesa lahko dostopajo le do svojega kosa pomnilnika. Pomnilnik drugih procesov jim je nedostopen (MMU).



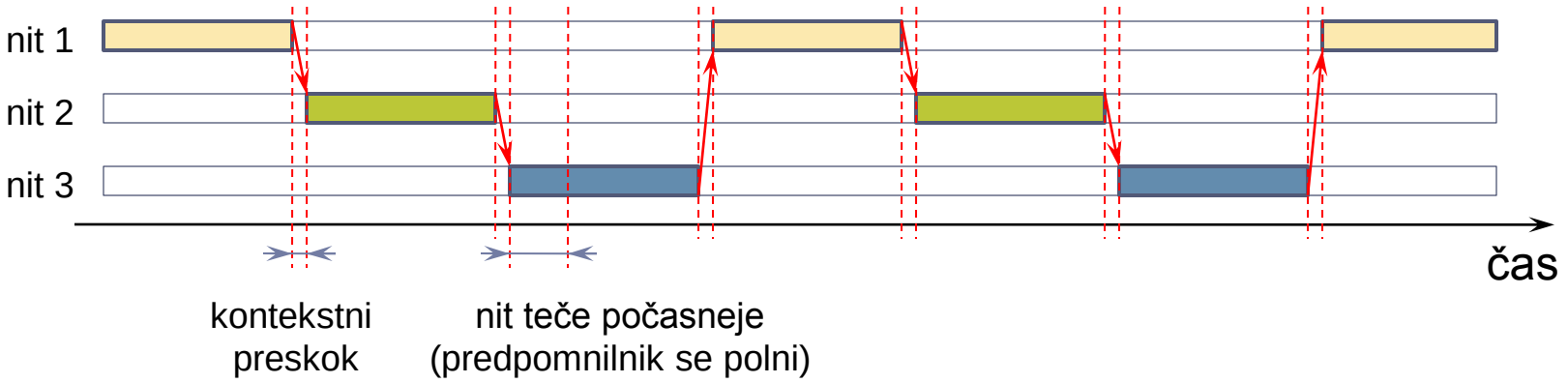
Izvedba večopravilnosti

- Hkrati lahko en procesor izvaja le eno nit.
  Izjema so SMT procesorji.
- Iluzijo večopravilnosti OS ustvari s hitrim preklapljanjem med nitmi, ki naj bi tekle vzporedno.
- Časovna os se razdeli na časovne rezine.
  V eni rezini se izvaja ena nit.



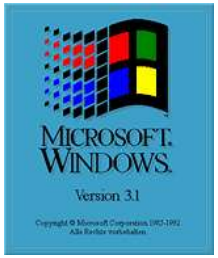
Kontekst niti in kontekstni preskok

- Kontekstni preskok (Context Switch) - preklop iz ene niti v drugo.
- Kontekst niti – stanje procesorja, ki pripadajo eni niti.
- Ob kontekstnem preskoku je potrebno shraniti kontekst niti, ki jo zapuščamo in naložiti (shranjen) kontekst niti v katero vstopamo.
- Kontekstni preskok lahko vzame veliko časa, saj obstaja precejšnja verjetnost, da mora po preskoku krmilnik predpomnilnika napolniti le-tega z novo vsebino, kar pa lahko traja nekaj časa. V tem času nit teče počasneje (imamo veliko “cache miss” dogodkov).



Kooperativna večopravilnost

- Kontekstni preskok se lahko zgodi ob trenutkih, ko program to dovoli s posebnim sistemskim kilcem.
- Dovolj velika pogostost kontekstnih preskokov (in s tem iluzija večopravilnosti) je odvisna od tega, ali posamezni programi dovolj pogosto dovolijo kontekstni preskok.
- En sam slabo napisan program povzroči, da se po njegovem zagonu izgubi iluzija večopravilnosti.
- Primer: Windows 3.1 (pred 1995), MAC OS (pred 2001).
- Še vedno popularna v vgrajenih sistemih s sprotnim odzivom (real-time ebedded systems), kjer je pomembna predvidljivost obnašanja.

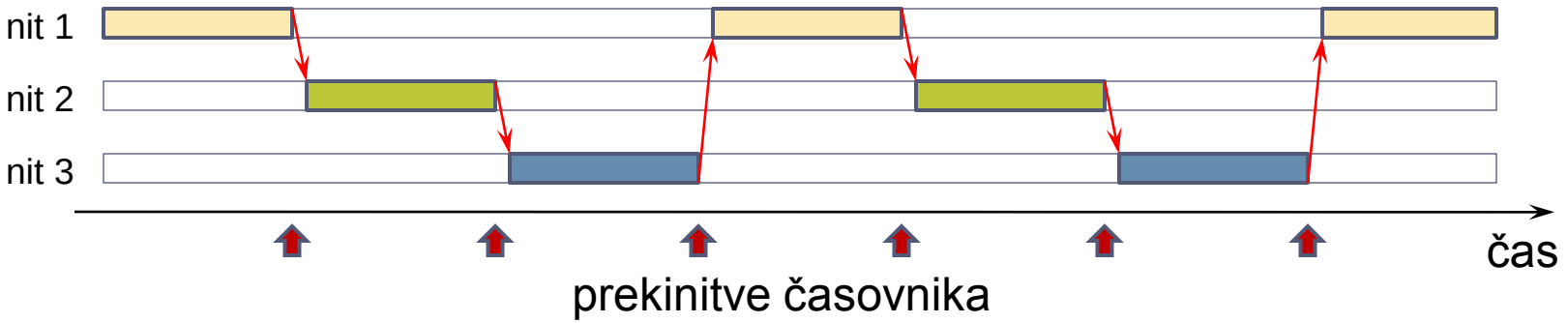


Predkupna večopravilnost
Preemptive Multitasking

- OS skrbi, da so kontekstni preskoki dovolj pogosti.
- Kontekstni preskok se lahko zgodi kadarkoli –
  program je lahko prekinjen kadarkoli.
- Kontekstni preskok se zgodi tudi takrat, ko program izvede sistemski klic in je prisiljen počakati na rezultate.
- Primer:
  Ko program želi brati podatke s sistemskim klicem read(), podatki pa še niso prispeli, se zgodi kontekstni preskok v drugi program. Ko podatki prispejo (sproži se prekinitev) se zgodi kontekstni preskok nazaj prvotni program, ki čaka na podatke.
- Iluzija večopravilnosti je zagotovljena brez, da bi zanjo moral skrbeti programer, ki piše programe za tak sistem.
- Pojavi se že v prvih verzijah sistemov UNIX (1969).
  Windows od različic NT 3.1 (1993) in 95 (1995) naprej.
  Linux od samega začetka.

Izvedba predkupnosti
Programiranje v predkupnem sistemu

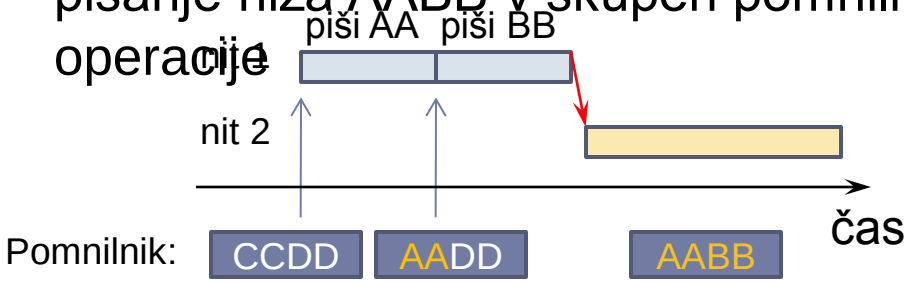
- Za kontekstne preskoke skrbi OS.
- Proženje kontekstnih preskokov se ponavadi izvede s pomočjo prekinitve časovnika (timer interrupt).
- Prekinitev časovnika je ponavadi prožena periodično.
- Prekinitveni podprogram izvede kontekstni preskok.



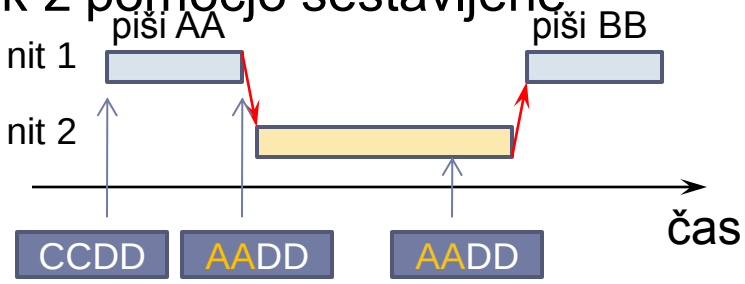
- Programerju ni treba skrbeti za kontekstne preskoke.
- Problem: komunikacija med nitmi.
- Problem: tvegana stanja.
- Problem: smrtni objem in stradanje.

Nedeljive (atomske) operacije in komunikacija med nitmi

- Nedeljive operacije so tako kratke, da se med njimi ne more zgoditi kontekstni preskok. Običajno so to operacije, ki ustrezajo enemu ukazu strojnega jezika mikroprocesorja.
- Nasprotje atomske operacije je sestavljena operacija.
- Kaj se zgodi, če imamo kontekstni preskok med sestavljeno operacijo?
- Primer:
  pisanje niza AABB v skupen pomnilnik z pomočjo sestavljene operacije



Nit 2 vidi novo vrednost (AABB)

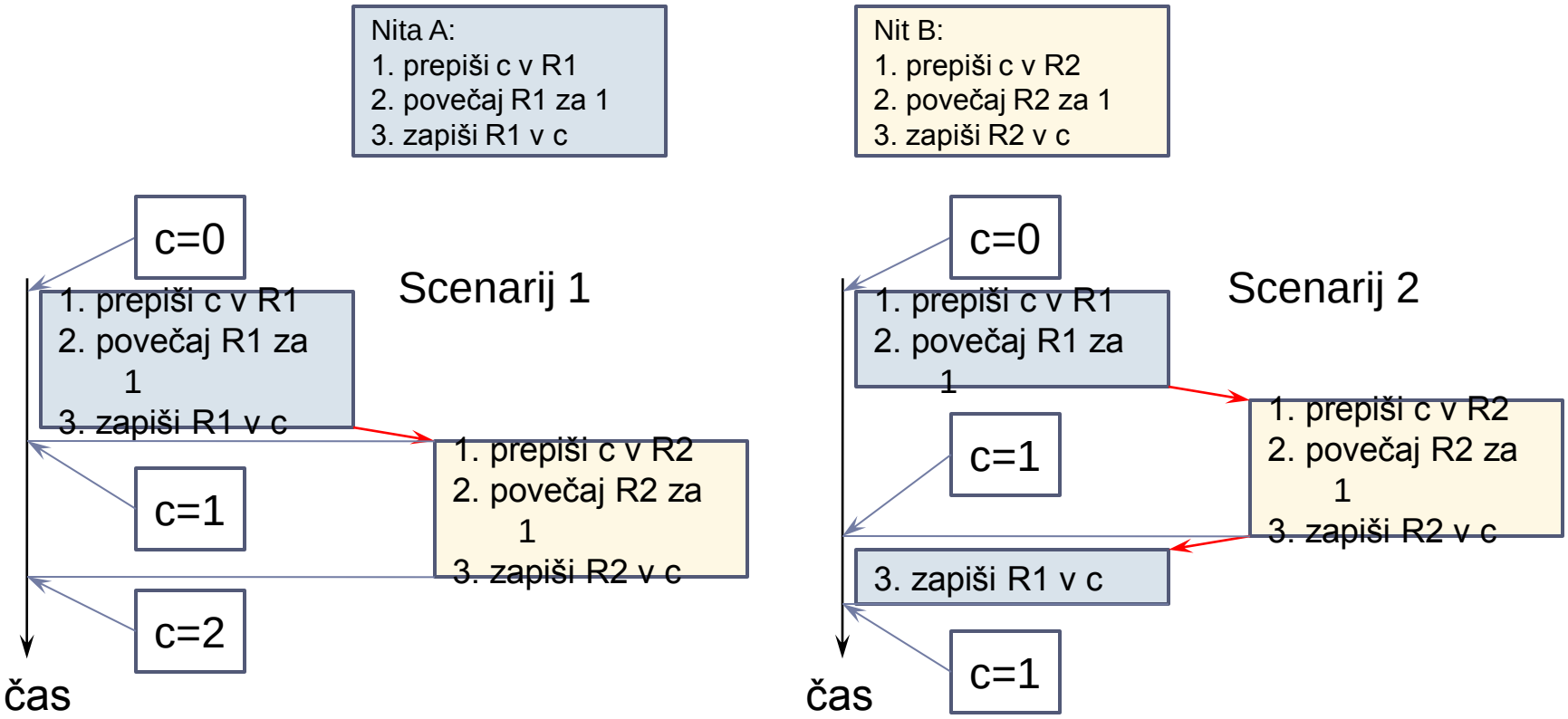


Nit 2 vidi neveljavno vrednost AADD,
(prva polovica je nova, druga polovica je stara)

- Če je pisanje niza AABB nedeljiva operacija, težav ni.

Tvegano stanje (race condition)

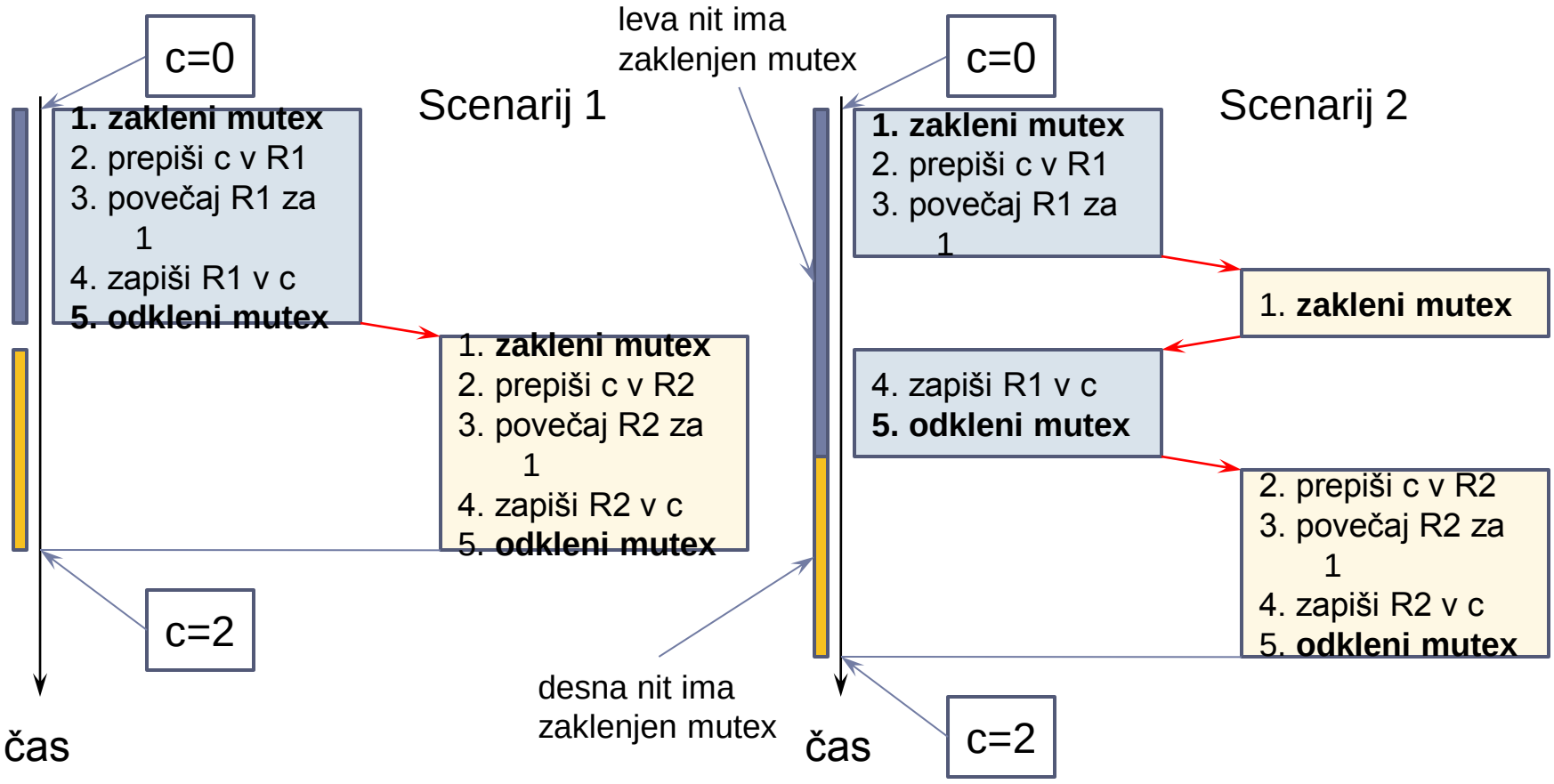
- Če je rezultat delovanja sistema odvisen od trenutka kontekstnega preskoka, pravimo da imamo tvegano stanje.
- Primer: vzporedno tečeta dve niti (A in B),
  ki želita obe povečati vrednost globalne spremenljivke c za 1.



Preprečevanje tveganih stanj
Mutex in semafor

- Mutex je objekt, ki ga lahko nit zaklene in odklene.
- Recimo, da je nit A zaklenila mutex.
- Če že zaklenjeni mutex poskuša zakleniti še kaka druga nit B, jo OS ustavi še preden ji uspe mutex zakleniti.
- Sledi kontekstni preskok iz niti B.
- Ko nit A odklene mutex, OS nadaljuje izvajanje niti B, ki sedaj lahko zaklene mutex.
- Mutex-e uporabljamo, da preprečimo navidezno vzporedno izvajanje delov programa, ki se ne smejo izvajati vzporedno.
- Tak del programa imenujemo kritični odsek (critical section).
- Semafor je Mutex, ki ga lahko n-krat zaklenemo preden je nadaljnje zaklepanje onemogočeno.

Primer uporabe mutex-a



- Ker je kritični odsek varovan s pomočjo mutex-a, do tveganega stanja ne more priti.

Smrtni objem (deadlock)

- Pogosto lahko neko enoto sistema (npr. trdi disk, serijska vrata, ...)
  uporablja le ena nit naenkrat.
- Hkratno uporabo lahko preprečimo z uporabo mutex-a.
  Imamo en mutex za vsako tako enoto.
- Če nit potrebuje enoto, zaklene njen mutex, ko konča, pa ga odklene.
- Recimo, da ima nit A rezervirano enoto E (zaklenjen mutex E), nit B pa rezervirano enoto F (zaklenjen mutex F).
- Nit A poskuša rezervirati enoto F... OS jo ustavi (nit B ima v rabi F).
- Nit B poskuša rezervirati enoto E... OS jo ustavi (nit A ima v rabi E).
- Nit A ne more sprostiti enote E, ker je nit A ustavljena.
- Nit B ne more sprostiti enote F, ker je nit B ustavljena.

Prednost (prioriteta) niti

- Pri dostopu do neke skupne enote sistema imajo lahko različne niti različno prednost (prioriteto).
- Več niti lahko hkrati čaka na pravico dostopa do skupne enote (na trenutek, ko bo mutex postal odklenjen).
- Ko se enota sprosti (mutex enote se odklene), OS dodeli enoto (dovoli zakleniti mutex) tisti niti, ki ima najvišjo prednost.
- Če imajo niti enako prednost, je prepuščeno OS-u, kateri niti bo kot naslednji dodelil enoto.

Stradanje (starvation)

- Recimo, da imamo tri niti (A, B in C).
- Najvišjo prednost ima nit A, najnižjo pa nit C.
- Nit A ima v uporabi enoto E.
- Če nit B želi uporabiti enoto, jo OS ustavi za čas dokler nit A enote ne sprosti.
- Ko je enota prosta, jo dobi nit B v uporabo.
- Če sedaj nit C potrebuje enoto, jo OS ustavi.
- Če se medtem tudi nit A “spomni”, da potrebuje enoto, jo OS ustavi.
- Ko nit B sprosti enoto, jo dobi v uporabo nit A, ker ima višjo prednost od niti C, čeprav se je “spomnila” tega pozneje kot nit C.
- Nit A in B si lahko tako izmenjujeta enoto v nedogled, nit C pa sploh ne dobi priložnosti za uporabo enote... **stradanje**.

Prioritetna inverzija (priority inversion)

- Tri niti: A (najvišja prioriteta), B in C (najnižja prioriteta).
- Nit C ima v rabi neko skupno enoto E.
- Nit A želi uporabiti enoto E.
- OS ustavi nit A, ker je E zasedena.
- Ker ima nit B višjo prioriteto, kot nit C, bo mikroprocesor medtem večino časa izvajal nit B, ne pa niti C.
- Nit C bo potrebovala veliko časa, da opravi svoje delo.
- Nit A bo morala dolgo čakati, da se enota E sprosti. Medtem se izvaja nit B, čeprav ima nižjo prioriteto kot nit A.

- Primer: težave sonde “Mars Pathfinder” ob pristajanju (1997).