

# Operacijski sistemi



Sočasnost

# Vsebina

- Sočasnost
- Sočasno izvajanje ukazov
- Souporaba vira
- Tvegano stanje
- Kritični odsek

# Sočasnost

- Osnovni pojmi
  - **sočasnost**, hkratnost (concurrency)
    - prekrivanje obstoja več procesov
    - občutek hkratnega izvajanja
      - menjavanje stanja: izvajanje in ne-izvajanje na procesorju
  - **vzporednost** (parallelism)
    - dejansko hkratno izvajanje (več procesorjev)
    - obstaja tudi vzporednost na nivoju procesorskih ukazov
    - posebna oblika sočasnosti
  - **porazdeljenost** (distributed)
    - izvajanje več procesov v več vozliščih omrežja



# Sočasnost

- Sočasnost in vzporednost
  - sočasnost brez vzporednosti
    - večopravilnost, prepletanje izvajanja ukazov
  - sočasnost z vzporednostjo
    - več-procesorski sistem, prekrivanje ukazov
  - vzporednost brez sočasnosti
    - vzporednost na nivoju ukazov
      - vzporedne bitne operacije ipd.
  - obe vrsti povzročita enake težave in rešitve
    - deljenje globalnih virov
    - smrtni objem (deadlock)
    - težavno odkrivanje programskih napak

# Sočasnost

- Kje se pojavlja **sočasnost**?
  - sočasno izvajanje več različnih aplikacij
    - multiprogramiranje je nastalo prav zaradi tega
  - **sočasno izvajanje ene aplikacije**
    - modularna zasnova in struktura aplikacij
    - pogled na aplikacijo kot množico vzporednih procesov
    - tudi sam OS lahko povzporejamo

# Sočasnost

- Odnosi med procesi
  - **tekmovalnost**
    - procesi se ne zavedajo drug drugega
    - tekmujejo med seboj za vire
      - npr. dve neodvisni aplikaciji želita tiskati
  - **sodelovanje preko medprocesne komunikacije**
    - procesi se **neposredno** zavedajo drug drugega
    - vzpostavijo medsebojno komunikacijo s komunikacijski mehanizmi, ki jih nudi OS
  - **sodelovanje preko souporabe virov**
    - procesi se **posredno** zavedajo drug drugega
    - komunikacija preko deljenega objekta

# Sočasno izvajanje ukazov

- **Prepletanje izvajanja**

- multiprogramiranje: več procesov, en procesor
- izvedba posameznih ukazov različnih procesov se poljubno prepleta

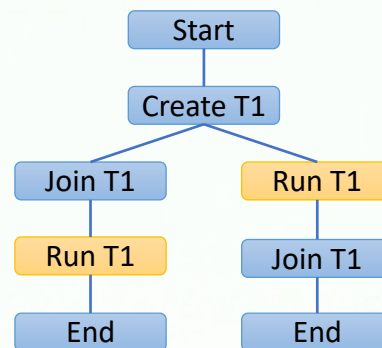
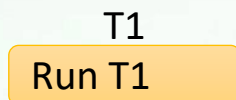
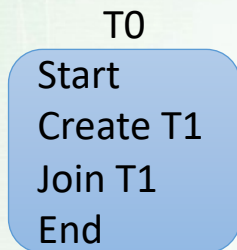
- **Prekrivanje izvajanja**

- multiprocesiranje: več procesov, več procesorjev
- izvedba ukazov različnih procesov se poljubno prekriva



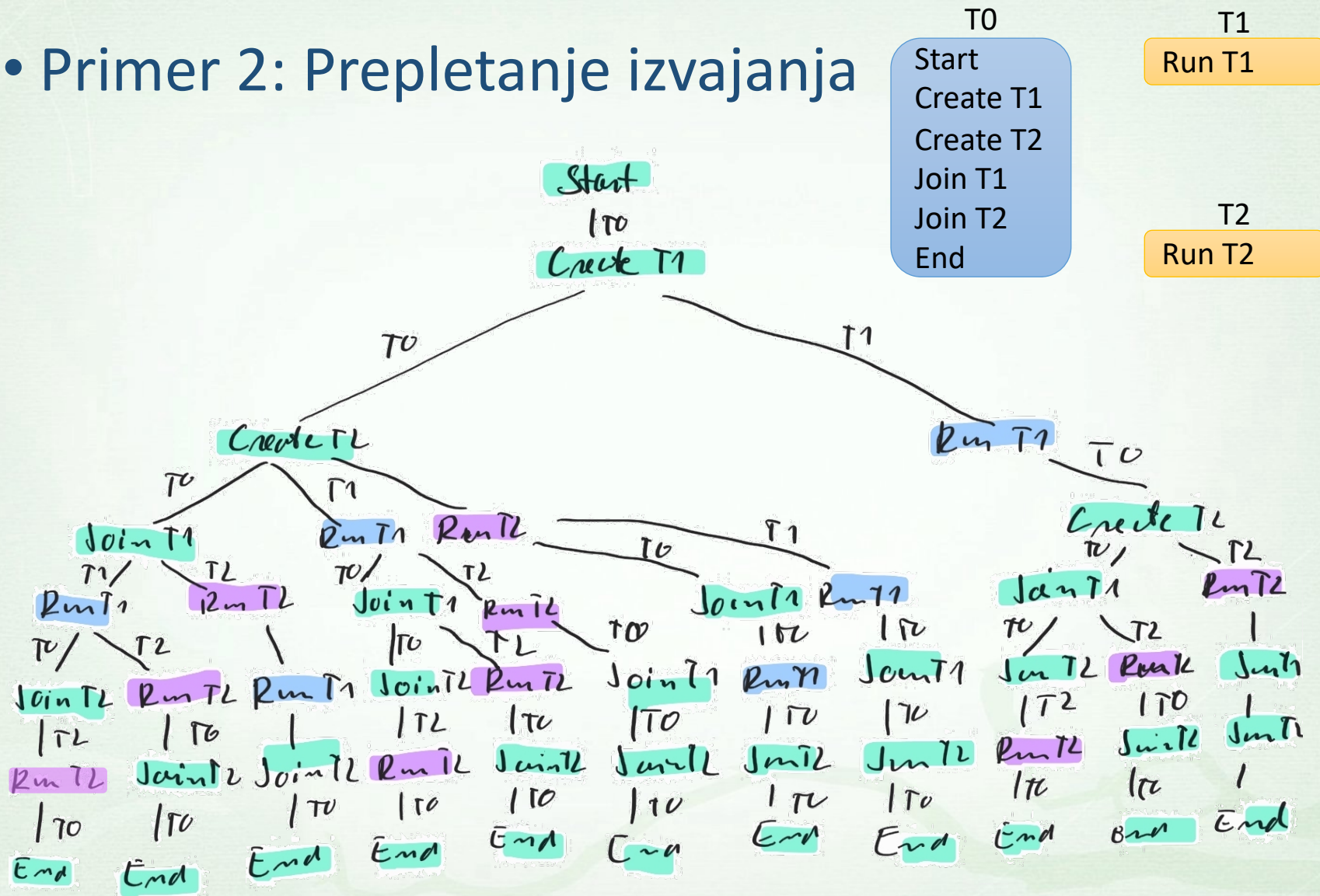
# Sočasno izvajanje ukazov

- Primer 1: Prepletanje izvajanja





- Primer 2: Prepletanje izvajanja



# Souporaba vira

- Skupni/deljeni vir (shared data)
  - vir, ki ga lahko uporablja dva ali več procesov
    - npr. podatki oz. spremenljivka v večnitnem programu
- Souporaba skupnega vira
  - sočasna uporaba vira s strani več procesov
  - rezultat izvajanja procesov
    - pričakovani: kar intuitivno pričakujemo
    - dejanski: rezultat, pridobljen z opazovanjem
      - nepričakovana vrednost vira
      - lahko celo različne vrednosti ob večkratnih zagonih pri “enakih” pogojih

# Souporaba vira

- Primer

- dve niti / procesa  $P1$  in  $P2$  se sočasno izvajata
- uporabljata skupni vir oz. spremenljivko
  - spremenljivka  $i$  z začetno vrednostjo  $i = 0$
  - eden inkrementira spremenljivko:  $i = i + 1$
  - drugi dekrementira spremenljivko:  $i = i - 1$
- opazujemo rezultat izvajanja
  - Kakšna je vrednost  $i$  po izvedbi obeh procesov?
    - pričakovana in dejanska
  - dejanska vrednost je odvisna od konkretne arhitekture procesorja in od kode v katero se prevede inc/dec



# Souporaba vira

DEMO: p. 277

- Primer

- V kakšno strojno kodo se program prevede?

load R, i  
add R, 1  
store R, i

load R, i  
sub R, 1  
store R, i

- Kako se prepleta izvajanje strojne kode?

P1  
load R, i  
add R, 1  
store R, i

P2

load R, i  
sub R, 1  
store R, i



i = ?

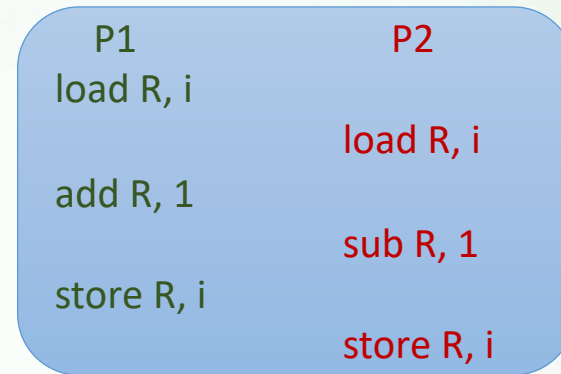
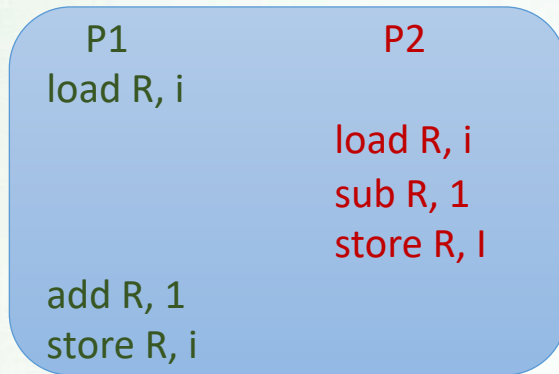




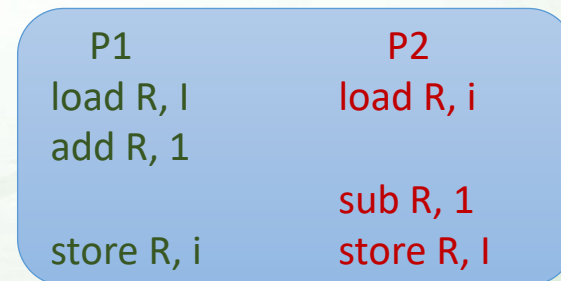
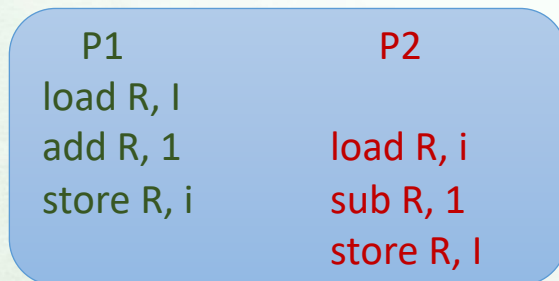
# Souporaba vira

- Primer

- poljubno **prepletanje** izvajanja ukazov



- poljubno **prekrivanje** izvajanja ukazov



# Tvegano stanje

- **Tvegano stanje** (race condition)
  - rezultat souporabe skupnega vira
    - odvisen od prepletanja/prekrivanja izvajanja ukazov
- **tveganim stanjem** se želimo izogniti
  - programska koda, ki povzroči tvegano stanje je nepravilna (programski hrošč)
  - izvajanje na različnih arhitekturah lahko različno pogosto vodi v tvegano stanje
    - hrošč lahko ostane neopazen tudi pri testiranju
    - težko razhroševanje sočasnih programov

# Tvegano stanje

- Še več primerov tveganega stanja

- dva procesa, en vir
- $P1: x = 1$
- $P2: x = 2$
- $x = ?$

- dva procesa, dva vira
- init:  $x = 1, y = 2$
- $P1: x = x + y$
- $P2: y = x + y$
- $x = ?, y = ?$



- $n$  procesov, en vir
- init:  $x = 0$
- $Pi: x = x + 1$
- $x = ?$

- dva procesa, dva vira
- init:  $x = 0, y = 0$
- $P1: x = x + 1, y = y - 1$
- $P2: x = x - 1, y = y + 1$
- $x = ?, y = ?$

# Kritični odsek

- **Kritični odsek** (critical section)
  - del programske kode, ki uporablja skupni vir
    - sočasna uporaba skupnega vira vodi v tvegano stanje
    - uporabo vira ščitimo s kritičnim odsekom
  - **ideja oz. rešitev**
    - le en proces naj bo sočasno v KO
    - ostalim procesom ne dovolimo vstopa dokler prvotni proces ne izstopi
    - princip **vzajemnega izključevanja**
    - **vstop** v in **izstop** iz kritičnega odsega

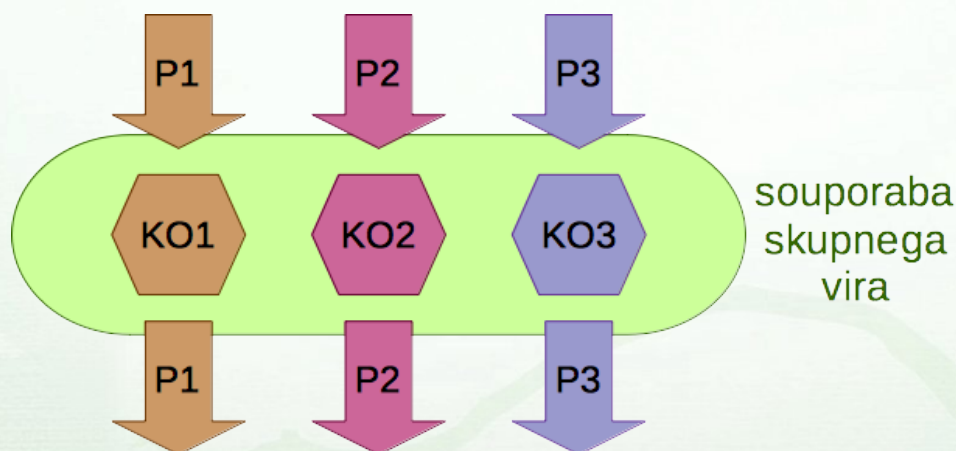




# Kritični odsek

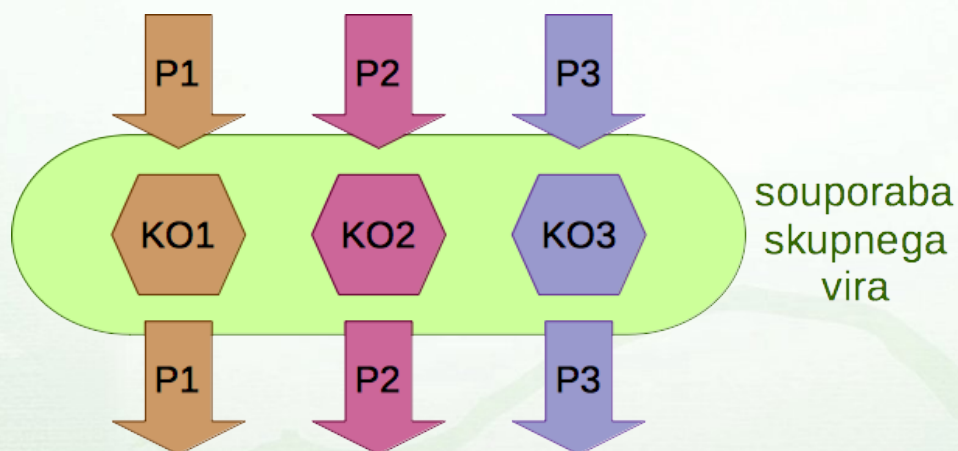


- Uporaba kritičnega odseka
  - **vstop** v kritični odsek
    - možen le, če noben drug proces ni v KO
      - **stanje** (pripravljen, izvajan ...) procesa **ni pomembno**, ampak le vrednost **programskega števca** (kje v programski kodi se proces nahaja)
  - po vstopu drugi procesi ne morejo v KO
    - kritični odsek je „zaseden“



# Kritični odsek

- Uporaba kritičnega odseka
  - **izstop** iz kritičnega odseka
    - vedno možen
    - po izstopu je KO „prost“
      - poljuben proces (lahko isti) lahko vstopi v KO



# Kritični odsek



- **Vzajemno izključevanje** (mutual exclusion)
  - mehanizem za preprečevanje tvegane stanja
    - zagotavlja, da se v KO nahaja kvečjemu en proces
    - preko vstopa in izstopa
- Je s tem problem kritičnega odseka razrešen?
  - tvegano stanje je rezrešeno
  - vendar se pojavijo nove težave in izzivi
    - težavi: stradanje, smrtni objem



# Kritični odsek

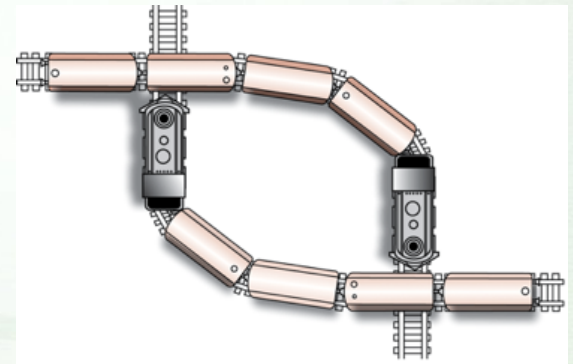
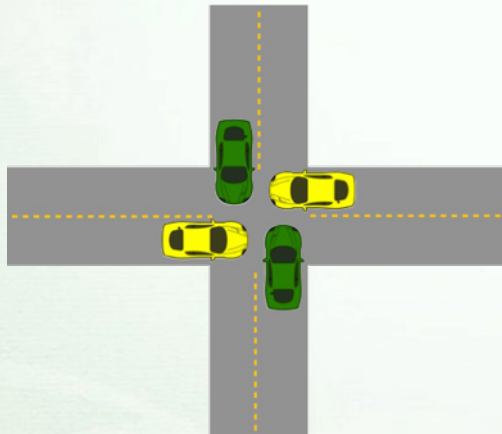
- Težava
  - **stradanje** (starvation)
    - proces, ki želi vstopiti v KO,  
ne pride na vrsto (ali pa predolgo čaka)
      - npr. vedno ga nekdo drug prehiti





# Kritični odsek

- Težava
  - **smrtni objem** (deadlock)
    - dva ali več procesov ne more nadaljevati, ker čakajo drug na drugega
      - npr. P1 in P2 oba potrebujeta R1 in R2 za nadaljevanje, pri tem P1 pridobi R1 in P2 pridobi R2



# Kritični odsek



- Izzivi
  - enakopravnost, poštenost
    - Je izbira procesa za vstop poštena? Kdo vpliva na izbor?
  - omejeno čakanje (bounded waiting)
    - vstop je odobren v končnem času (izogibanje stradanju)
  - učinkovitost
    - hitrost izbor procesa, vstop in izstop
  - poljubna hitrost izvajanja procesov
    - poljubno prepletanje in/ali prekrivanje ukazov
  - splošnost rešitve
    - za poljubno število procesov