

Primjer 1. Simulacija Petersonova algoritma

```
uđi_u_KO (I) {  
    J = 1 - I  
    ZASTAVICA[I] = 1  
    PRAVO = J  
    dok je ZASTAVICA[J] == 1 && PRAVO == J radi  
        ;  
}  
  
izadi_iz_KO (I) {  
    ZASTAVICA[I] = 0  
}
```

Simulacija Petersonova algoritma (1)

Dretva D0 u svom NKO, dretva D1 želi ući u KO:

D1: poziva Uđi_u_KO(1)

D1: ZASTAVICA[1] = 1

D1: PRAVO = 0

D1: dok je (**ZASTAVICA[0] == 1** && **PRAVO == 0**) //uvjet nije zadovoljen

D1: izlazi iz funkcije Uđi_u_KO => ulazi u svoj KO

Simulacija Petersonova algoritma (2)

Dretva D1 u svom KO, dretva D0 želi ući u KO:

D0: poziva Uđi_u_KO(0)

D0: ZASTAVICA[0] = 1

D0: PRAVO = 1

D0: dok je (ZASTAVICA[1] == 1 && PRAVO == 1) //uvjet je zadovoljen

D0 vrti petlju dok D1 ne izađe iz KO i postavi svoju zastavicu na 0

D1: poziva Izađi_iz_KO(1) => postavlja ZASTAVICA[1] = 0

D0: izlazi iz petlje, izlazi iz funkcije Uđi_u_KO => ulazi u svoj KO

Simulacija Petersonova algoritma (3)

Dretva D1 i D2 istovremeno žele ući u KO:

D0: poziva Uđi_u_KO(0)

D1: poziva Uđi_u_KO(1)

D0: ZASTAVICA[0] = 1

D1: ZASTAVICA[1] = 1

D0: PRAVO = 1

D1: PRAVO = 0

D0: dok je (ZASTAVICA[1] == 1 && PRAVO == 1) //uvjet nije zadovoljen

D1: dok je (ZASTAVICA[0] == 1 && PRAVO == 0) //uvjet je zadovoljen

D1 vrti petlju dok D0 ne izađe iz KO i postavi svoju zastavicu na 0

D0: izlazi iz petlje, izlazi iz funkcije Uđi_u_KO => ulazi u svoj KO

D0: poziva Izađi_iz_KO(0) => postavlja ZASTAVICA[0] = 0

D1: izlazi iz petlje, izlazi iz funkcije Uđi_u_KO => ulazi u svoj KO

Primjer 2. U nekom sustavu pokrenut je proces koji se sastoji od **X dretvi** istog oblika:

```
Dretva () {
```

```
    za i=1 do 1000 {
```

```
        A=A+1;
```

```
    }
```

```
}
```

↓
dohvati A u reg.
povećaj taj reg. za 1
spremi reg. u A

Varijabla A je zajednička i početna vrijednost joj je 0.
Koja je "očekivana", koja vjerojatna a koja minimalna moguća vrijednost varijable A nakon što sve dretve obave svoj posao, ako je:

	„očekivana”	vjerojatna	minimalna
a) X = 1	1000	1000	1000
b) X = 2	2000	≤ 2000	2
c) X = 3	3000	≤ 3000	2
d) X = N	$N \cdot 1000$	$\leq N \cdot 1000$	2

„Vjerojatni” scenarij za prethodni primjer

Dretva D1 i D2 izvode petlju, trenutna vrijednost $A = x$:

1: D1: pročitaj A u $\text{Reg-}y = x$ ($\text{Reg-}y$ je dio konteksta dretve D1)

2: D2: pročitaj A u $\text{Reg-}y = x$ ($\text{Reg-}y$ je dio konteksta dretve D2)

3: D1: povećaj $\text{Reg-}y = x + 1$

4: D2: povećaj $\text{Reg-}y = x + 1$

5: D1: spremi $\text{Reg-}y$ u A ($A=x+1$)

6: D2: spremi $\text{Reg-}y$ u A ($A=x+1$)

Ukupno su dretve obavile dvije iteracije, ali A je povećan samo za 1

Takvih iteracija može biti puno pa je konačna vrijednost od A i značajno manja od očekivane

„Najgori” scenarij za prethodni primjer

Problem je što se u operacijskom sustavu (koji raspoređuje dretve) može svašta dogoditi. Npr. prekid, pojava neke druge prioritetnije dretve (osim ovih koje razmatramo) i slično.

Takvi događaji prekidaju pojedine dretve i mogu rezultirati „vrlo čudnim” redoslijedom izvođenja dretvi (koje razmatramo).

Stoga je „najgori slučaj”, koji je opisan na idućem slajdu, zaista jako malo vjerojatan, ali jest moguć.

Nešto što nam na prvu izgleda kao malo vjerojatno, možda biti i vrlo vjerojatno (ipak ne opisani slučaj). Naime, dretve obično vrlo brzo obavljaju neku petlju i obavljaju ju JAKO puno puta. Neka sasvim mala vjerojatnost se tako značajno može povećati.

„Najgori” scenarij za prethodni primjer

Dretva D1 i D2 kreću s radom, trenutna vrijednost $A = 0$:

1: D1: pročitaj A u $\text{Reg-}y = 0$

2: D2: pročitaj A u $\text{Reg-}y = 0$ i odradi sve osim zadnje iteracije ($A=N-1$)

3: D1: povećaj $\text{Reg-}y = 0 + 1$

4: D1: spremi $\text{Reg-}y$ u A ($A=1$)

5: D2: pročitaj A u $\text{Reg-}y = 1$ (zadnja iteracija za D2)

6: D1: odradi sve i završi s radom ($A=N$)

7: D2: povećaj $\text{Reg-}y = 1 + 1 = 2$

8: D2: spremi $\text{Reg-}y$ u A ($A=2$)

9: D2: završi s radom

U slučaju više dretvi, one mogu sve odraditi prije 3. trenutka