

Lamportov postupak međusobnog isključivanja

Uz pomoć Lamportovog algoritma moguće je sinkronizirati dvije ili više dretvi, odnosno dva ili više procesa koji primjerice mogu izgledaju ovako:

```
proces proc(i){           /* i  [0..n] */
    ponavljaj{
        uđi u kritični odsječak;
        K.O.;
        izađi iz kritičnog odsjeka;
        N.K.O.;
    }dok je zadovoljen(uvjet);
}
```

Lamportov algoritam:

zajedničke varijable: ULAZ[0..**n**-1], BROJ[0..**n**-1]

```
funkcija uđi_u_kritični_odsječak(i)
{
    ULAZ[i] = 1;
    BROJ[i] = max(BROJ[0],...,BROJ[n-1]) + 1;
    ULAZ[i] = 0;

    za j = 0 do n-1 čini
        dok je ULAZ[j] <> 0 čini
            ništa;
        dok je BROJ[j] <> 0 && (BROJ[j] < BROJ[i] || (BROJ[j] == BROJ[i] && j < i)) čini
            ništa;
}
```

```
funkcija izađi_iz_kritičnog_odsjeka(i)
{
    BROJ[i] = 0;
}
```

Dodatne upute:

Ako se program rješava s procesima tada treba zajedničke varijable tako organizirati da se prostor za njih zauzme odjednom i podijeli među njima. Ovo je nužno zbog ograničenog broja segmenata i velikog broja korisnika.

Ovisno o opterećenju računala i broju procesa koji se pokreću, a da bi se vidjele razlike prilikom izvođenja programa može se usporiti izvršavanje programa sa:

```
sleep(1);
```

na kraju kritičnog odsječka (K.O.).

Problemi zbog izvođenja instrukcija "preko reda"

Za ispravan rad Dekkerovog, Petersonovog i Lamportovog algoritma pretpostavlja se da se instrukcije programa izvode zadanim redoslijedom. Međutim, neki procesori, kako bi ubrzali izvođenje programa, dozvoljavaju izvođenje instrukcija i "preko reda" (engl. *out-of-order*). Takvo ponašanje može uzrokovati greške u algoritmu međusobnog isključivanja.

Primjerice, prilikom izvođenja Lamportovog algoritma može se dogoditi da se sljedeći niz instrukcija ne izvodi zadanim redoslijedom.

```
1:  ULAZ[i] = 1;  
2:  BROJ[i] = max(BROJ[0], ..., BROJ[n-1]) + 1;  
3:  ULAZ[i] = 0;
```

Procesor koji omogućuje izvođenje instrukcija "preko reda" bi nakon prve linije koda mogao ustanoviti da su instrukcije u drugoj i trećoj liniji međusobno nezavisne i mogao bi ih izvesti proizvoljnim redoslijedom, npr. liniju 3 prije linije 2 (razlog za to može biti da se element `ULAZ[i]` nalazi u priručnom spremniku, a `BROJ[i]` ne). U tom se slučaju ne radi više o Lamportovom algoritmu.

Kako bi se izbjeglo proizvoljno izvođenje instrukcija može se jezičnom procesoru naznačiti da je pristup spornim varijablama točno određen programskim kodom na način da se varijabla proglasi tipom podataka `atomic`:

```
#include <stdatomic.h>  
atomic_int ULAZ[N], BROJ[N];
```