Problem pet filozofa

Zadatak

S pomoću više dretvi riješiti problem pet filozofa koristeći koncept monitora. Pri svakoj promjeni program mora vizualno prikazati za sve filozofe što oni rade. Npr. kada filozof 4 ide jesti, tada treba ispis izgledati otprilike ovako: "Stanje filozofa: X o O X o" (X-jede, O-razmišlja, o-čeka na vilice).

Problem pet filozofa. Filozofi obavljaju samo dvije različite aktivnosti: misle ili jedu. To rade na poseban način. Na jednom okruglom stolu nalazi se pet tanjura te pet štapića (između svaka dva tanjura po jedan). Filozof prilazi stolu, uzima lijevi štapić, pa desni te jede. Zatim vraća štapiće na stol i odlazi misliti.
Ako rad filozofa predstavimo jednim zadatkom onda se on može opisati na sljedeći način:

```
filozof i
    ponavljati
        misliti;
        jesti;
    do zauvijek;
```

Slika 5.1. Pseudokod dretve filozof

Potrebno je pravilno sinkronizirati rad filozofa. Uporabom binarnog semafora za pojedine štapiće može doći do potpunog zastoja (kada svi istovremeno uzmu lijevi štapić). Problem se može riješiti uvođenjem dodatnog općeg semafora koji će ograničavati broj filozofa za stolom. Međutim, u općem slučaju kada zadaci dijele više sredstava i koriste ih više istovremeno semafori mogu uzrokovati potpuni zastoj. Upravo zbog takvih problema dijeljenja više sredstava među više zadataka uvodi se novi mehanizam za sinkronizaciju: monitori.
Monitor se sastoji od skupa procedura i strukture podataka nad kojima procedure djeluju i koje nisu vidljive izvan monitora. Procedure su tako konstruirane da djeluju samo jedna po jedna . Unutar monitora ispituje se uvjet koji utječe na odvijanje zadatka. Ukoliko je uvjet ispunjen, zadatak zauzima sredstva i obavlja potrebne akcije. Po završetku oslobađa sredstva te dozvoljava drugom zadatku ulazak u monitor te izlazi iz monitora. Ukoliko po ulasku u monitor uvjet nije ispunjen tada zadatak odlazi u red čekanja na taj uvjet, tj. prividno napušta monitor.
Za ostvarenje monitora u višedretvenom programu stoje na raspolaganju funkcije međusobnog isključavanja te funkcije za rukovanje uvjetnim varijablama koje služe za ostvarenje reda uvjeta.

Struktura rješenja

Pseudokod na slici 5.1 pretvoriti u dio programa koji radi svaka dretva tako da `misliti` i `jesti` postaju funkcije. Funkcija `misliti()` treba simulirati trošenje vremena (npr. `sleep(3)`). Funkcija `jesti(n)` simulira postupak hranjenja filozofa s uključivo potrebnim kodom sinkronizacije koji je skiciran u nastavku.

```
jesti(n){
   /* n - redni broj filozofa */
   uđi_u_kritični_odsječak;
      filozof[n] = 'o';
      dok (vilica[n] == 0 || vilica[(n + 1) % 5] == 0)
         čekaj_u_redu_uvjeta(red[n]);
      vilica[n] = vilica[(n + 1) % 5] = 0;
      filozof[n] = 'X';
      ispiši_stanje(n);
   izađi_iz_kritičnog_odsječka;

   njam_njam; /* sleep(2); */

   uđi_u_kritični_odsječak;
      filozof[n] = 'O';
```

```
            vilica[n] = vilica[(n + 1) % 5] = 1;
            oslobodi_dretvu_iz_reda(red[(n - 1) % 5]);
            oslobodi_dretvu_iz_reda(red[(n + 1) % 5]);
            ispiši_stanje(n);
         izađi_iz_kritičnog_odsječka;
      }
```

Primjer ispisa pokretanja programa

```
# ./a.out
X 0 0 0 0 (1)
X o X 0 0 (3)
O o X o o (1)
O o X o X (5)
O o O o X (3)
O X O o X (2)
O X O o O (5)
O O O o O (2)
O O O X O (4)
X O O X O (1)
X O o O O (4)
X O X O O (3)
O O X O O (1)
O O X O X (5)
O o O O X (3)
O X O O X (2)
O X O o O (5)
O X O X O (4)
o O O X O (2)
X O O X O (1)
X O o O O (4)
X O X O O (3)
^C
#
```