Paralelno programiranje

Prva domaća zadaća

Zadatak: uporabom MPI-a izraditi simulaciju raspodijeljenog problema n filozofa.

Slično uobičajenoj inačici problema, model obuhvaća *n* filozofa koji sjede za okruglim stolom na kojemu se nalazi hrana. Na raspolaganju je *n* vilica, od kojih za potrebe prehrane svaki filozof mora koristiti dvije. Svaki filozof koristi točno određene vilice (njegovu 'lijevu' i njegovu 'desnu'), tj. svaku pojedinačnu vilicu mogu koristiti samo dva susjedna filozofa. Za razliku od uobičajenog rješenja sa zajedničkim spremnikom, filozofi moraju biti implementirani kao procesi koji komuniciraju isključivo razmjenom poruka (raspodijeljena okolina). Iz istog razloga, vilice ne stoje na stolu nego se uvijek nalaze kod nekog od filozofa (procesa). Program se mora moći pokrenuti u proizvoljnom broju procesa ($n > 1$), a ispis je potrebno prilagoditi tako da svaki proces/filozof ispisuje promjene stanja uz uvlačenje teksta (*tabs*) proporcionalno indeksu procesa.

```
Proces(i)
{   misli (slucajan broj sekundi);        // ispis: mislim
    nabavi vilice;                        // ispis: trazim vilicu (indeks)
    jedi (slucajan broj sekundi);         // ispis: jedem
}
```

Algoritam koji treba implementirati opisan je u radu K.M.Chandy, J. Misra: "The drinking philosophers problem" (https://www.fer.unizg.hr/_download/repository/Philosophers.pdf) (poglavlje 4), a ukratko je objašnjen u nastavku.

Svaka od *n* vilica može biti *čista* ili *prljava*, te se u jednom trenutku može nalaziti samo kod jednog filozofa (naravno). Na početku, sve su vilice *prljave*. Također, vilice su na početku podijeljene tako da se svaka vilica, koju mogu dijeliti dva susjedna filozofa, nalazi kod filozofa s nižim rednim brojem (indeksom procesa). Slijedom navedenog, filozof s indeksom 0 na početku ima dvije vilice, a filozof s indeksom *n*-1 niti jednu. Svi filozofi slijede ova pravila:

1. *filozof jede ako je gladan i ako ima obje vilice (bez obzira jesu li čiste ili prljave ;)*
2. nakon jela, obje korištene vilice postaju prljave.
3. ako filozof želi jesti, šalje zahtjeve za vilicom koja nije kod njega i čeka na odgovor.
4. ako filozof ne jede, a postoji zahtjev za *prljavu* vilicu koja se nalazi kod njega, vilicu čisti i šalje je susjedu.
5. svaki filozof pamti zahtjeve za vilicama koje je dobio od svojih susjeda.
6. dok filozof misli, udovoljava svakom zahtjevu za vilicom koja je trenutno kod njega.

Iz navedenih pravila je vidljivo da filozof ne udovoljava zahtjevu za *čistom* vilicom - zahtjev će biti udovoljen tek kad vilica postane prljava (nakon jela). Isto tako, ukoliko filozof misli (trenutno nije gladan), obvezan je odmah (što prije) udovoljiti zahtjevima drugih filozofa.

```
Proces(i)
{   misli (slucajan broj sekundi);                      // ispis: mislim
    i 'istovremeno' odgovaraj na zahtjeve!              // asinkrono, s povremenom provjerom
    dok (nemam obje vilice) {
        posalji zahtjev za vilicom;                     // ispis: trazim vilicu (i)
        ponavljaj {
            cekaj poruku (bilo koju!);
            ako je poruka odgovor na zahtjev            // dobio vilicu
                azuriraj vilice;
            inace ako je poruka zahtjev                 // drugi traze moju vilicu
                obradi zahtjev (odobri ili zabiljezi);
        } dok ne dobijes trazenu vilicu;
    }
    jedi;                                               // ispis: jedem
    odgovori na postojeće zahtjeve;                     // ako ih je bilo
}
```

"Istovremeno" odgovaranje na zahtjeve potrebno je izvesti tako da filozof povremeno provjerava postoji li neki zahtjev (nema poruka) upućen njemu. Ova funkcionalnost može se postići funkcijom *MPI_Iprobe* koja trenutno (neblokirajuće) provjerava postoji li neka poruka upućena promatranom procesu (predavanja, poglavlje 2.8). Tek nakon što funkcija detektira postojanje dolazne poruke, ista se treba primiti s *MPI_Recv*. U našem primjeru, interval provjeravanja postojanja dolaznih poruka nije kritičan i može biti jednak 1 sekundu.
