DEO VI: UPRAVLJANJE DELJENIM RESURSIMA

DEO VI: UPRAVLJANJE DELJENIM RESURSIMA

Nakon odslušanog ovog dela, trebalo bi da možete da:

- objasnite pojam upravljanja deljenim resursima;
- navedete i objasnite modele upravljanja deljenim resursima;
- objasnite probleme nadmetanja za deljene resurse;
- objasnite utrkivanje;
- objasnite mrtvo blokiranje;
- objasnite živo blokiranje;
- objasnite izgladnjivanje;
- navedete i objasnite načine rešavanja mrtvog blokiranja.

BLOKIRANJE

Primer dva procesa koji se uzajamno blokiraju čekanjem na poruku od onog drugog

P1	P2
...	...
Receive (P2);	Receive (P1);
...	...
Send (P2, M1);	Send (P1, M2);

MODELI PRISTUPA DELJENIM RESURSIMA

- U teoriji i praksi konkurentnog programiranja koristi se nekoliko modela (test-primera) za ispitivanje i demonstraciju praktično svakog novopredloženog koncepta za sinhronizaciju i komunikaciju između procesa
- Ovi modeli odslikavaju tipične situacije nadmetanja konkurentnih procesa za pristup do deljenih resursa koje se sreću pri konstrukciji OS i konkurentnih programa
- Standardni modeli:
  - *ograničeni bafer* (*bounded buffer*) – već obrađen
  - *čitaoci i pisci* (*readers-writers*)
  - *filozofi koji večeraju* (*dining philosophers*)

ČITAOCI I PISCI

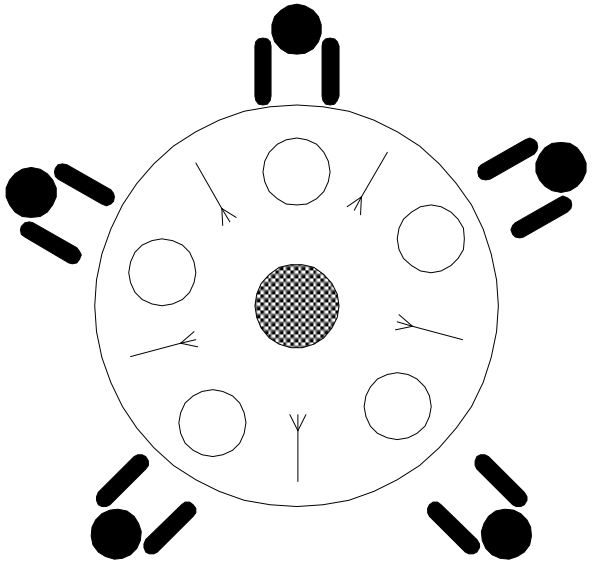
- Koncept monitora obezbeđuje međusobno isključenje pristupa konkurentnih procesa do deljenog resursa, uz eventualnu uslovnu sinhronizaciju. Međutim, koncept potpunog međusobnog isključenja kod monitora ponekad predstavlja suviše restriktivnu politiku koja smanjuje konkurentnost programa
- Često se operacije nad deljenim resursom mogu svrstati u operacije koje:
  - samo čitaju deljene podatke, odnosno ne menjaju stanje resursa (operacije čitanja)
  - upisuju u deljene podatke, tj. menjaju stanje resursa (operacije upisa)

  Koncept monitora ne dozvoljava nikakvu konkurentnost ovih operacija

FILOZOFI KOJI VEČERAJU

- Problem *filozofa koji večeraju* (*dining philosophers*, Dijkstra 1965.):

  Pet filozofa sedi za okruglim stolom na kome se nalazi posuda sa špagetima, jedu i razmišljaju. Svaki filozof ima svoj tanjir, a između svaka dva susedna tanjira stoji po jedna viljuška. Pretpostavlja se da su svakom filozofu, da bi se poslužio, potrebne dve viljuške, kao i da može da koristi samo one koje se nalaze levo i desno od njegovog tanjira. Ako je jedna od njih zauzeta, on mora da čeka. Svaki filozof ciklično jede, pa razmišlja. Kad završi sa jelom, filozof spušta obe viljuške na sto i nastavlja da razmišlja. Posle nekog vremena, filozof ogladni i ponovo pokušava da jede. Potrebno je definisati protokol (pravila ponašanja, algoritam) koji će obezbediti ovakvo ponašanje filozofa i pristup do viljušaka



UTRKIVANJE

- Primer (nekorektne) uslovne sinhronizacije:
  - *suspend*: bezuslovno suspenduje pozivajući proces
  - *resume*: bezuslovno deblokira imenovani proces, ako je blokiran
- Ovakav neispravan uslov naziva se *utrkivanje* (*race condition*). Nastaje zbog preplitanja delova koji bi morali biti izvršeni neprekidivo
- Tipično nastaje kao posledica toga što se odluka o promeni stanja procesa (suspenziji) donosi na osnovu ispitivanja vrednosti deljene promenljive, pri čemu ta dva koraka nisu nedeljiva, pa može doći do preuzimanja, tj. "utrkivanja" od strane drugog procesa koji pristupa istoj deljenoj promenljivoj

MRTVO BLOKIRANJE

- Scenario: svaki filozof uzme svoju levu viljušku i čeka na desnu?!
- Ovakav neregularan uslov nastaje tako što se grupa procesa koji konkurišu za deljene resurse međusobno kružno blokiraju - *mrtvo* (ili *kružno*) *blokiranje* (*deadlock*)
- U opštem slučaju, mrtvo blokiranje nastaje tako što se grupa procesa nadmeće za ograničene resurse, pri čemu proces *P1* drži ekskluzivan pristup do resursa *R1* i pri tom čeka blokiran da se oslobodi resurs *R2*, proces *P2* drži ekskluzivan pristup do resursa *R2* i pri tom čeka blokiran da se oslobodi resurs *R3*, itd., proces *Pn* drži ekskluzivan pristup do resursa *Rn* i pri tom čeka blokiran da se oslobodi resurs *R1*. Tako procesi ostaju neograničeno suspendovani u cikličnom lancu blokiranja

ŽIVO BLOKIRANJE

- Scenario: svi filozofi uzmu svoju levu viljušku, ne mogu da uzmu desnu, pa spuste levu, i tako ciklično!?
- Ovakva neregularna situacija u konkurentnom programu, kod koje se grupa procesa izvršava, ali nijedan ne može da napreduje jer u petlji čeka na neki uslov, naziva se *živo blokiranje* (*livelock*)
- Treba razlikovati živo od mrtvog blokiranja. Iako se u oba slučaja procesi nalaze "zaglavljeni" čekajući na ispunjenje nekog uslova, kod mrtvog blokiranja su oni suspendovani, dok se kod živog izvršavaju, tj. uposleno čekaju
- Obe situacije su neispravna stanja jer nije obezbeđena njegova živost (*liveness*)

IZGLADNJIVANJE

- Scenario: Filozof X, njegov levi L, desni D; u jednom trenutku L može da uzme obe svoje viljuške, što sprečava filozofa X da uzme svoju levu viljušku; pre nego što L spusti svoje viljuške, D može da uzme svoje, što opet sprečava filozofa X da počne da jede; teorijski, ovaj postupak se može beskonačno ponavljati, što znači da filozof X nikako ne uspeva da zauzme svoje viljuške (resurse) i počne da jede, jer njegovi susedi naizmenično uzimaju njegovu levu, odnosno desnu viljušku!?
- Ovakva neregularna situacija u konkurentnom programu, kod koje jedan proces ne može da dođe do željenog resursa jer ga drugi procesi neprekidno pretiču i zauzimaju te resurse, naziva se *izgladnjivanje* (*starvation*), ili *neograničeno odlaganje* (*indefinite postponement*), ili *lockout*
- Drugi primer: prikazano rešenje čitalaca i pisaca; ko izgladnjuje?