Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (IR3OS2)
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika
Kolokvijum: Prvi, decembar 2014.
Datum: 6.12.2014.

Prvi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 1,5 sat. Dozvoljeno je korišćenje literature.

Zadatak 1 _____/10 *Zadatak 3* _____/10
Zadatak 2 _____/10

Ukupno: _____/30 = _____%

Napomena: Ukoliko u zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku, da je uokviri (da bi se lakše prepoznala prilikom ocenjivanja) i da nastavi da izgrađuje preostali deo svog odgovora na temeljima uvedene pretpostavke. Ocenjivanje unutar potpitanja je po sistemu "sve ili ništa", odnosno nema parcijalnih poena. Kod pitanja koja imaju ponuđene odgovore treba **samo zaokružiti** jedan odgovor. Na ostala pitanja odgovarati **čitko, kratko i precizno**.

1. (10 poena) Raspoređivanje procesa

U nekom sistemu koristi se *Multilevel Feedback-Queue Scheduling* (MFQS) na sledeći način:

- Postoje tri reda spremnih procesa: HP (*High Priority*), MP (*Medium Priority*) i LP (*Low Priority*).
- Globalni algoritam raspoređivanja je po prioritetu, s tim da HP ima najviši, a LP najniži prioritet.
- Raspoređivanje u svim redovima je *Round-Robin* (RR), samo sa različitim vremenskim kvantumom koji se dodeljuje procesima.
- Procesima koji se uzimaju iz HP dodeljuje se vremenski kvantum 2, onima koji se uzimaju iz MP vremenskim kvantum 4, a onima iz LP kvantum 8.
- Proces koji je tek postao spreman (bio je blokiran ili je tek kreiran) smešta se u prvi viši red od onoga iz koga je otišao u stanje blokade (u HP ako je otišao iz HP ili MP, u MP ako je otišao iz LP), odnosno u HP ako je prvi put aktiviran.
- Proces kome je istekao vremenski kvantum smešta se u MP ako je prethodno bio uzet iz HP, odnosno u LP ako je prethodno bio uzet iz MP ili LP.

Posmatra se jedan proces koji ima sledeće nalete izvršavanja (označeni sa C i dužinom trajanja naleta) i ulazno/izlazne operacije (označene sa I/O):

C7, I/O, C3, I/O, C1, I/O, C7, I/O, C5

Dati oznake redova spremnih procesa (HP, MP, LP) u koje je ovaj proces redom stavljan, i to za svako stavljanje procesa u neki od redova spremnih (odgovor dati u obliku npr. HP, MP, LP, LP, LP, ...)

Odgovor: _____

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Korišćenjem klasičinih monitora i uslovnih promenljivih, realizovati monitor `TaxiDispatcher` koji implementira sledeće ponašanje dispečera taksija. Korisnik (*user*) i taksi vozilo (*taxi*) su procesi koji se prijavljuju dispečeru pozivom procedura `userRequest` i `taxiAvailable`, respektivno.

Korisnik poziva proceduru `userRequest` kada želi da dobije vožnju taksijem. Ako trenutno ima raspoloživih taksi vozila koja čekaju u redu na zahtev korisnika, korisniku će odmah biti dodeljeno jedno od tih raspoloživih vozila. U suprotnom, korisnik će biti blokiran i čekaće u redu korisnika sve dok mu dispečer ne dodeli vozilo. Broj dodeljenog taksi vozila korisnik dobija u izlaznom parametru `assignedTaxiID`.

Taksi vozač poziva proceduru `taxiAvailable` kada je slobodan da primi i preveze korisnika. Kao ulazni parametar `myID` dostavlja svoj broj vozila. Ako trenutno ima korisnika koji čekaju u redu na raspoloživo vozilo, tom vozilu će biti dodeljen jedan od tih korisnika. U suprotnom, taksi će biti blokiran i čekaće u redu raspoloživih vozila sve dok mu dispečer ne dodeli korisnika.

Semantika uslovnih promenljivih je takva da proces koji je čekao na uslovnoj promenljivoj i oslobođen je ima prioritet u nastavku izvršavanja u odnosu na proces koji je signalizirao tu uslovnu promenljivu. Oba ta procesa, naravno, imaju prioritet u odnosu na druge procese koji čekaju na ulaz u monitor.

```
monitor TaxiDispatcher;
  export userRequest, taxiAvailable;

  var ...;

  procedure userRequest (var assignedTaxiID : integer); ...

  procedure taxiAvailable (myID : integer); ...

begin ... end;
```

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Na programskom jeziku Java napisati serverski proces koji implementira sledeće ponašanje dispečera taksija. Korisnik (*user*) i taksi vozilo (*taxi*) su udaljeni procesi koji se prijavljuju dispečeru porukama `userRequest` i `taxiAvailable`, respektivno. Korisnik šalje poruku `userRequest` kada želi da dobije vožnju taksijem. Ako trenutno ima raspoloživih taksi vozila koja čekaju u redu na zahtev korisnika, korisniku će odmah biti dodeljeno jedno od raspoloživih vozila. U suprotnom, korisnik će biti blokiran i čekaće u redu korisnika sve dok mu dispečer ne dodeli vozilo. Taksi vozač šalje poruku `taxiAvailable` kada je slobodan da primi i preveze korisnika. Ako trenutno ima korisnika koji čekaju u redu na raspoloživo vozilo, tom vozilu će biti dodeljen jedan od tih korisnika. U suprotnom, taksi će biti blokiran i čekaće u redu raspoloživih vozila sve dok mu dispečer ne dodeli korisnika. Prilikom oslobađanja korisnika i taksi vozila treba ispoštovati isti redosled kao prilikom njihovog pristizanja. Serverski proces „osluškuje“ port 1033 preko koga prima zahteve. Na raspolaganju je generička klasa `LinkedList <T>` koja realizuje red uređen po redosledu umetanja. Ova klasa nije predviđena za konkurentno pozivanje (*thread-safe*) i ima standardne metode `add(T)` i `T remove()` za smeštanje na kraj, odnosno uzimanje jednog elementa sa početka reda, kao i proveru da li je red prazan `boolean isEmpty()`. Međuprocesnu komunikaciju realizovati preko priključnica (*socket*) i razmenom poruka (*message passing*). Nije potrebno proveravati uspešnost izvršavanja operacija (*try/catch* klauzule).

Rešenje: