Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (SI3OS2)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo
Kolokvijum: Prvi, oktobar 2014.
Datum: 1.11.2014.

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):

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

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.

Taksi vozač poziva proceduru `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.

```
monitor TaxiDispatcher;
  export userRequest, taxiAvailable;

  var ...;

  procedure userRequest (); ...

  procedure taxiAvailable (); ...

begin ... end;
```

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Potrebno je realizovati serverski proces na programskom jeziku Java koji ima mogućnost prihvatanja SQL upita i izvršavanja nad lokalnom bazom podataka. SQL upit se prosleđuje kao jedan string, za koji smatrati da ne sadrži znak `'#'` u sebi. Pored sadržaja upita, prilikom svakog zahteva prosleđuje se i maksimalno vreme čekanja za taj upit, pa poruka koja predstavlja jedan zahtev je u sledećem formatu "`#sqlQuery#waitTime#`". Klijentski proces zadaje jedan ovakav zahtev za izvršavanje nakon čega čeka na odgovor. Zahtevi se smeštaju u dva reda, jedan uređenu po vremenu pristizanja zahteva, a drugi uređen po vremenu isteka roka čekanja. Za izvršavanje se bira zahtev sa početka drugog reda, ako je njegovo vreme isteklo, u suprotnom, uzima se zahtev sa kraja prvog reda. U kontekstu serverskog procesa svaki zahtev izvršava se u vidu jedne niti tipa `SQLWorkerThread` čiji je interfejs i deo realizacije dat u nastavku.

```
public class SQLWorkerThread extends Thread implements Delayed{
 protected long delayTime=0;
 protected string sqlQuery, result;
 protected PrintWriter outStream;
 protected boolean finished = false;

 public SQLWorkerThread(string sqlQuery, int waitTime, PrintWriter
outStream) {
 delayTime = System.currentTimeMillis()+waitTime;
 //...
 }
 public void run() {
  String result = //execute sqlQuery
  outStream.println(result);
 }
 public long getDelay() {
  return delayTime-System.currentTimeMillis();
 }
}
```

Argument `sqlQuery` u konstruktoru označava sadržaj upita, `waitTime` zadati rok čekanja, a `outStream` izlazni tok priključnice po kojoj je zahtev poslat. U kontekstu serverskog procesa posebna nit zadužena je za raspoređivanje pristiglih zahteva i započinjanje izvršavanja. Za realizaciju skupa zahteva na raspolaganju je klasa `Queue<SQLWorkerThread>` uređena po redosledu umetanja, kao i klasa `DelayedQueue<SQLWorkerThread>` koja je uređena po vremenu isticanja roka svakog elementa, a na osnovu rezultata metoda `getDelay()`. Ove klase nisu predviđene za konkurentno pozivanje (*thread-safe*) i imaju standardne metode `add(SQLWorkerThread)` i `remove(SQLWorkerThread)` za smeštanje odnosno uzimanje jednog elementa iz reda, dohvatanje elementa sa početka bez izbacivanja `SQLWorkerThread peek()`i proveru da li je red prazan `boolean isEmpty()`. Međuprocesnu komunikaciju realizovati preko priključnica (*socket*) i razmenom poruka (*message passing*). Dozvoljeno je korišćenje koda prikazanog na vežbama (kod sa vežbi ne treba prepisivati, nego precizno navesti koja klasa ili koji metod se koriste i/ili menjaju, nasleđuju, ...). Nije potrebno proveravati uspešnost izvršavanja operacija (*try/catch* klauzule).

Rešenje: