Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (SI3OS2, IR3OS2)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Prvi, septembar 2015.
Datum: 25.8.2015.

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 klasa `Scheduler`, čiji je interfejs dat dole, realizuje raspoređivač spremnih procesa po sledećoj varijanti MFQS (engl. *multi-level feedback queue scheduling*), tako da i operacija dodavanja novog spremnog procesa `put()` i operacija uzimanja spremnog procesa koji je na redu za izvršavanje `get()` imaju ograničeno vreme izvršavanja koje ne zavisi od broja spremnih procesa (kompleksnost $O(1)$):

- Postoje dva reda spremnih procesa, od kojih je jedan „aktivan“, a drugi „pasivan“.
- Proces za izvršavanje se uzima iz aktivnog reda po *FCFS* redosledu.
- U operaciji `put()` drugi argument ukazuje na to da li proces koji se dodaje u skup spremnih dolazi iz stanja suspenzije (`wasBlocked==1`), ili iz stanja spremnosti, pošto mu je isteklo dodeljeno vreme izvršavanja (`wasBlocked==0`).
- Proces koji u skup spremnih dolazi iz stanja suspenzije ili je tek aktiviran (`wasBlocked==1`) stavlja se u aktivan red. Sami odredite u koji red treba smeštati procese kojima je isteklo vreme izvršavanja.
- Kada se aktivni red isprazni, aktivni i pasivni redovi zamenjuju svoje uloge (aktivni postaje pasivni i obratno).
- Pretpostaviti da uvek ima spremnih procesa, tj. da nikada nisu oba reda prazna.
- U strukturi `PCB` postoji polje `next` kao pokazivač tipa `PCB*` koji služi za ulančavanje struktura PCB u jednostruke liste.

Realizovati u potpunosti klasu `Scheduler`.

```
class Scheduler {
public:
  Scheduler ();
  PCB* get ();
  void put (PCB* pcb, int wasBlocked);
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Korišćenjem klasičnih uslovnih promenljivih, napisati kod za monitor koji ima dve operacije, *flip* i *flop*, uz sledeću sinhronizaciju: između dva susedna izvršavanja operacije *flip* mora se najmanje *N* puta izvršiti operacija *flop*. Drugim rečima, nakon jednog izvršavanja *flip*, mora doći najmanje *N* (može i više) izvršavanja *flop*, pa onda opet može *flip* itd.

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Na programskom jeziku Java napisati serverski proces koji implementira sledeće ponašanje predsednika licitacije. Korisnici (*user*) su udaljeni procesi koji se prijavljuju predsedniku porukama `userRequest`. Korisnik šalje poruku `userRequest` kada želi da prisustvuje licitaciji. Prijavljenom korisniku će biti poslata trenutna cena tekućeg kruga licitacije. Kada primi cenu, korisnik šalje odgovor. Odgovor može biti nova cena ili informacija da neće povećavati cenu u ovom krugu. Korisnik nakon toga čeka blokiran sve dok ne dobije neku poruku od predsednika. Predsednik prikuplja sve cene koje korisnici pošalju u toku jednog kruga. Jedan krug traje dok svi korisnici ne pošalju svoje odgovore. Ako u jednom krugu nije bilo nijedne nove cene, licitacija se završava i korisniku koji je dao najveću cenu u prethodnom krugu šalje se informacija o kupovini, dok se ostalim šalje informacija o završetku licitacije. Ako je bilo nove cene, svim korisnicima se šalje ta nova cena i počinje novi krug licitacije. Serverski proces „osluškuje“ port 1033 preko koga prima zahteve. Međuprocesnu komunikaciju realizovati preko priključnica (*socket*). Nije potrebno proveravati uspešnost izvršavanja operacija (*try*/*catch* klauzule).

Rešenje: