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 2012.
Datum: 21.8.2012.

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	*Zadatak 4*	_____/10

Ukupno: _____/40 = _____%

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, implementira raspoređivač spremnih procesa po prioritetu (engl. *priority scheduling*). Implementirati ovu klasu tako da i operacija dodavanja novog spremnog procesa `put()` i operacija uzimanja spremnog procesa koji je na redu za izvršavanje `get()` budu ograničene po vremenu izvršavanja vremenom koje ne zavisi od broja spremnih procesa (kompleksnost $O(1)$). Između spremnih procesa sa istim prioritetom raspoređivanje treba da bude *FCFS*. Konstanta `MAXPRI` je maksimalna vrednost prioriteta (prioriteti su u opsegu `0..MAXPRI`, s tim da je 0 najviši prioritet). U slučaju da nema drugih spremnih procesa, treba vratiti proces na koga ukazuje `idle` (to je uvek spreman proces najnižeg prioriteta). U strukturi `PCB` polje `priority` tipa `int` predstavlja prioritet procesa, a polje `next` pokazivač tipa `PCB*` koji služi za ulančavanje struktura PCB u jednostruke liste.

```
const int MAXPRI = …;
extern PCB* idle;

class Scheduler {
public:
  Scheduler ();
  PCB* get ();
  void put (PCB*);
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Monitor `server`, čiji je interfejs dat dole, služi kao jednoelementni bafer za razmenu podataka između proizvoljno mnogo uporednih procesa proizvođača koji pozivaju operaciju `write` i potrošača koji pozivaju operaciju `read`. Tek kada jedan proizvođač upiše podatak tipa `Data` operacijom `write`, jedan potrošač može da ga pročita operacijom `read`; tek nakon toga neki proizvođač sme da upiše novi podatak, i tako dalje naizmenično. Implementirati monitor `server` sa potrebnom sinhronizacijom korišćenjem standardnih uslovnih promenljivih.

```
type Data = ...;

monitor server;
  export write, read;
  procedure write (data : Data);
  procedure read (var data : Data);
end;
```

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Projektuje se konkurentni klijent-server sistem koji koristiti priključnice (engl. *sockets*) i mehanizam prosleđivanja poruka (engl. *message passing*). Klijenti su procesi koji ciklično obavljaju svoje aktivnosti. Pre nego što u jednom ciklusu neki klijent započne svoju aktivnost, dužan je da od servera traži dozvolu u obliku "žetona" (engl. *token*). Kada dobije žeton, klijent započinje aktivnost. Po završetku aktivnosti, klijent vraća žeton serveru. Server „osluškuje“ port 1033 preko koga prima zahteve i vodi računa da u jednom trenutku ne može biti izdato više od N žetona: ukoliko klijent traži žeton, a ne može da ga dobije jer je već izdato N žetona, klijent se blokira. Napisati kod procesa-servera i procesa-klijenta. Nije potrebno proveravati uspešnost izvršavanja operacija nad priključnicama.

Rešenje:

4. (10 poena) Upravljanje deljenim resursima

U nekom sistemu su tri procesa (P_1, P_2, P_3) i tri različite instance resursa (R_1, R_2, R_3). Ne primenjuje se izbegavanje mrtve blokade, već se samo prati zauzeće resursa pomoću grafa, a resurs dodeljuje procesu čim je slobodan. Procesi su izvršili sledeće operacije datim redosledom:

P2.request(R3), P1.request(R1), P2.request(R2), P3.request(R3), P2.release(R3)

a)(5) Nacrtati graf zauzeća resursa nakon ove sekvence.

b)(5) Navesti neki nastavak ove sekvence koji sistem vodi u mrtvu blokadu (*deadlock*).

Rešenje: