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 2015.
Datum: 23.10.2015.

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`, čija je delimična definicija data dole, realizuje raspoređivač spremnih procesa primenjujući jednu aproksimaciju *SJB* (engl. *shortest job first*) algoritma, 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 $O(1)$):

- Postoji N redova spremnih procesa, pri čemu je N konfiguraciona konstanta. Red sa nižim indeksom ima viši prioritet (tj. red sa indeksom 0 je najvišeg prioriteta).
- U PCB procesa postoji polje `tau` koje predstavlja procenu dužine sledećeg naleta izvršavanja (engl. *CPU burst*). Ova vrednost izračunata je pre nego što kernel pozove operaciju `Scheduler::put` da bi stavio proces u red spremnih.
- Postoji niz od N konstanti `tresholds` koji sadrži pragove za dužinu procene `tau`. Proces se smešta u red najvišeg prioriteta `p` za koji je `tau<=thresholds[p]`. Na taj način, procesi sa kraćom procenjenom dužinom narednog naleta imaju viši prioritet.
- Za izvršavanje se bira proces iz reda sa najvišim prioritetom, a unutar istog reda po *FCFS* redosledu.
- Ukoliko su svi redovi prazni, operacija `Scheduler::get` treba da vrati 0.
- 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*);
private:
  static unsigned thresholds[N];
  static const int N;
  ...
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Klasa `Server`, čiji je interfejs dat dole, služi kao jednoelementni bafer za razmenu podataka između proizvoljno mnogo uporednih niti proizvođača koji pozivaju operaciju `put` i potrošača koji pozivaju operaciju `get`. Tek kada jedan proizvođač upiše podatak tipa `Data` operacijom `put`, jedan potrošač može da ga pročita operacijom `get`; tek nakon toga neki proizvođač sme da upiše novi podatak, i tako dalje naizmenično. Na jeziku Java implementirati klasu `Server` sa potrebnom sinhronizacijom.

```
class Server {
  public void put (Data d);
  public Data get ();
}
```

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Na programskom jeziku Java implementirati server za balansiranje opterećenja. Prilikom kreiranja servera zadaje se niz radnih stanica. Za svaku raspoloživu radnu stanicu u nizu se nalazi jedan string sa IP adresom radne stanice i portom na kojem radna stanica očekuje zahteve (adresa i port su odvojeni dvotačkom). Po pokretanju, server treba da osluškuje port 6000 kako bi prihvatao zahteve od klijenata. Kada klijent uspostavi konekciju, šalje string "request" i očekuje string sa adresom i portom radne stanice. Po pristizanju odgovora, konekcija se raskide, nakon čega se klijent obraća radnoj stanici. Po završetku obrade, radna stanica uspostavlja konekciju ka serveru preko porta 6001, nakon čega šalje serveru svoju adresu (isti string koji je na serveru zapamćen) i broj zahteva koje trenutno obrađuje i raskida vezu. Adresa i broj zahteva su odvojeni znakom '#'. Server vodi evidenciju o brojevima zahteva koje radne stanice obrađuju kako bi svaki zahtev klijenta preusmerio na najmanje opterećenu radnu stanicu. Nakon slanja odgovora klijentu, broj zahteva za dodeljenu radnu stanicu se uvećava za 1. Svaki put kada od radne stanice stigne informacija o broju zahteva koje obrađuje, taj broj se pamti umesto broja koji je za tu radnu stanicu prethodno bio zapisan na serveru (bez obzira na to što se može desiti da neki klijent koji je preusmeren na tu radnu stanicu možda još nije poslao zahtev radnoj stanici).

Rešenje: