Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Računarska tehnika i informatika, Softversko inženjerstvo
Kolokvijum: Prvi, januar 2018.
Datum: 15. 1. 2018.

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 za simetrični multiprocesorski sistem sa tzv. *afinititetom* (engl. *affinity*) procesa: teži se da dati proces dobije isti procesor na kom se prethodno već izvršavao, kako bi se smanjilo kašnjenje zbog promašaja u procesorskom kešu koje bi postojalo kada bi se on izvršavao na drugom procesoru. Osim toga, sistem dodeljuje dati procesor onom procesu dodeljenom tom procesoru, koji je od početka svog tekućeg naleta izvršavanja (tj. od kako je došao u red spremnih npr. iz stanja suspenzije) imao najkraće ukupno vreme izvršavanja na procesoru tokom tekućeg naleta izvršavanja (engl. *CPU burst*).

- Postoji *P* simetričnih (jednakih i ravnopravnih) procesora, pri čemu je *P* konfiguraciona konstanta.
- Za svaki od *P* procesora postoji poseban red spremnih procesa, uređenih po ukupnom vremenu izvršavanja procesa na procesoru u tekućem naletu izvršavanja; ovo vreme čuva se u polju `execTime` strukture PCB svakog procesa.
- Kada stavlja dati proces u red spremnih operacijom `Scheduler::put()`, u drugom argumentu `elapsed` ove operacije, sistem dostavlja vreme koje se proces izvršavao na procesoru pre nego što ga je izgubio; ukoliko je proces izgubio procesor zbog isteka vremenskog kvanta ili nekog prekida, ova vrednost jednaka je stvarnom vremenu koje je proteklo od kada je proces dobio procesor; ukoliko pak proces dolazi iz stanja suspenzije ili je tek aktiviran, ova vrednost je 0.
- Ako proces dolazi u red spremnih jer je izgubio procesor (`elapsed!=0`), stavlja se u red spremnih istog procesora na kom se već izvršavao. Taj procesor zapisan je u polju `affinity` strukture PCB.
- Ako proces dolazi u red spremnih iz stanja suspenzije ili je tek aktiviran (`elapsed==0`), stavlja se u red spremnih onog procesora koji ima najmanje spremnih procesa u svom redu, radi raspodele opterećenja procesora (engl. *load balancing*).
- Ukoliko je red spremnih procesa prazan, 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 (int proc);  // Get the process for the given processor
  void put (PCB*, unsigned long elapsed);
private:
  static const int P; // Number of processors
  ...
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Na jeziku Java napisati kod za monitor `Toggle` koji ima dve operacije, `flip` i `flop`, a koje pozivaoci mogu izvršavati strogo naizmenično (prvo `flip`, pa onda `flop` itd.).

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Napisati na programskom jeziku Java server koji sinhronizuje klijente. Klijenti imaju mogućnost da se prijave serveru, pri čemu prijava može biti prihvaćena ili odbijena, i imaju mogućnost da jave serveru da su završili posao. Server prihvata nove klijente sve dok bar jedan klijent ne završi posao. Nakon toga server ne dozvoljava prijavu novih korisnika sve dok svi prijavljeni korisnici ne završe sa radom. Protokol komunikacije između klijenata i servera osmisliti po potrebi. Nije potrebno prepisivati kod dat na vežbama, samo precizno navesti šta i gde je korišćeno.

Rešenje: