Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo
Kolokvijum: Prvi, novembar 2023.
Datum: 06. 11. 2023.

Prvi kolokvijum iz Operativnih sistema 2

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. 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 raspoređivanje procesa po prioritetima i MFQS na sledeći način. Svaki proces u polju `defPri` svog PCB ima svoj podrazumevani prioritet zadat pri pokretanju, u opsegu od 0 do `MAX_PRI` (niža vrednost je viši prioritet). Tekući prioritet u polju `pri` je dinamički prioritet koji se može menjati samo u okviru margina `defPri`±`PRI_MARGIN`, gde je `PRI_MARGIN` konstantna margina. Kada proces dolazi u red spremnih iz stanja suspenzije (parametar `wasBlocked=true` operacije `Scheduler::put`), tekući prioritet mu se povećava (vrednost `pri` se smanjuje) za 1; kada procesu istekne vremenski odsečak i on se ponovo stavlja u red spremnih (parametar `wasBlocked=false`), prioritet mu se smanjuje (vrednost `pri` se povećava) za 1, sve u okviru pomenutih granica.

Postoje tri reda spremnih procesa, red višeg prioriteta HP, red srednjeg prioriteta MP i red nižeg prioriteta LP. Procesi sa vrednošću `pri` manjom od `HP_PRI` smeštaju se u red HP; inače, procesi sa vrednošću `pri` manjom od `MP_PRI` smeštaju se u red MP, ostali se smeštaju u red LP. Na taj način se proces vremenom degradira ili promoviše, u zavisnosti od toga da li u red spremnih dolazi iz stanja supenzije ili zbog isteka vremenskog odsečka. Proces za izvršavanje uzima se iz HP, ako takav postoji; inače iz MP, ako takav postoji; inače iz LP. Raspoređivanje u svakom redu je *Round Robin*.

Klasa `Scheduler`, čiji je interfejs dat dole, realizuje opisani raspoređivač spremnih procesa. Implementirati u potpunosti 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 $O(1)$). U slučaju da nema drugih spremnih procesa, operacija `get()` vraća *null*. Polje `next` strukture `PCB` služi za ulančavanje u liste.

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

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Napraviti monitor koji svojim operacijama *startRead*, *stopRead*, *startWrite* i *stopWrite* obezbeđuje protokol „više čitalaca – jedan pisac“ (engl. *multiple readers – single writer*), ali tako da prednost daje čitaocima (i time potencijalno izgladnjuje pisce) na sledeći način:

- ako su neki čitaoci već u toku operacije čitanja, naredni čitaoci se puštaju bez čekanja;
- kada pisac završi sa operacijom upisa, pušta se naredni pisac, ako čeka, u suprotnom se puštaju svi čitaoci koji čekaju.

Koristiti klasične uslovne promenljive proširene dopunjene sledećim operacijama:

- *signalAll*: deblokira sve procese koji čekaju na uslovnoj promenljivoj i jednog po jednog pušta da nastavi svoju započetu operaciju monitora pre ostalih procesa;
- *waiting*: funkcija koja vraća broj procesa koji čekaju na uslovnoj promenljivoj.

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Napisati kod servera koji omogućava prijavljivanje korisnika. Korisnik ima korisničko ime, šifru i broj telefona. Korisnik može da se registruje na serveru slanjem svojih podataka. Nakon uspešnog prijavljivanja može da koristi server. Prijavljivanje se vrši u dva koraka. U prvom koraku korisnik šalje svoje korisničko ime i šifru koju je koristio prilikom registracije. U slučaju neispravnih podataka prijavljivanje se odbija. Nakon toga server šalje korisniku slučajan ceo broj putem mobilnog telefona. Korisnik treba da taj broj pošalje serveru. Ako se broj poklapa sa onim što je poslato putem mobilnog telefona, prijavljivanje je uspešno obavljeno. Korisnik može da pokušava proizvoljan broj puta da se prijavi, a kada uspešno završi prijavljivanje može da od servera traži željene usluge (obradu usluga nije potrebno implementirati, samo staviti komentar na mesto na kom bi taj kod bio). Server treba da ima mogućnost obrade zahteva od više klijenata uporedo. Server osluškuje na portu 5555.

U okviru servera dostupne su sledeće funkcije:

```
void sendSms(String phoneNumber, int code);
```

koja šalje ceo broj putem mobilnog telefona i

```
int generateRadomCode();
```

koja generiše nasumičan ceo broj.

Rešenje: