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 2020.
Datum: 06. 11. 2020.

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 koristi se raspoređivanje procesa sa prioritetom u opsegu `0..MaxPri` (niža vrednost označava viši prioritet). Kako bi se raspoređivanje približilo ponašanju SJF, prioritet se određuje na osnovu procene dužine sledećeg naleta izvršavanja. U skupu procesa istog prioriteta, raspoređivanje je *round-robin*. U strukturi `PCB` postoje sledeća polja:

- Polje `time` tipa `unsigned long` predstavlja vreme proteklo u izvršavanju procesa (ukupno vreme korišćenja procesora u jednom naletu izvršavanja); svaki put kada procesu oduzme procesor, jezgro (van klase `Scheduler`) na ovu vrednost (kumulativno) dodaje vreme proteklo od kada je proces poslednji put dobio procesor (pre eventualnog poziva `Scheduler::put()`). Ovu vrednost treba resetovati kada proces postaje spreman nakon što je bio blokiran (u funkciji `Scheduler::put(wasBlocked==1)`).
- Polje `tau` predstavlja procenu dužine sledećeg naleta izvršavanja; ovu vrednost treba postaviti na novi procenu kada proces postaje spreman nakon što je bio blokiran (u funkciji `Scheduler::put(wasBlocked==1)`), a pre smeštanja u odgovarajući red. Pri kreiranju procesa polja `tau` i `time` su postavljena na unapred definisanu vrednost.
- Polje `next` je pokazivač na sledeći PCB u listi (za ulančavanje u jednostruke liste).

Procena dužine sledećeg naleta izvršavanja radi se eksponencijalnim usrednjavanjem sa koeficijentom 1/2. Prioritet procesa određuje se na osnovu tako procenjenog naleta na sledeći način: ako je `tau` manje od konstante `TAU`, prioritet je 0; ako je veće ili jednako od `TAU` i manje od `2*TAU`, prioritet je 1 itd, ako je veće ili jednako `MaxPri*TAU`, prioritet je `MaxPri`.
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*. Drugi argument operacije `put()` govori da li je proces postao spreman zato što je prethodno bio blokiran (`wasBlocked=1`) ili mu je istekao vremenski kvantum (`wasBlocked=0`).

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

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Korišćenjem klasičnih uslovnih promenljivih implementirati monitor `ResourcePool` koji realizuje rezervoar (*pool*) raspoloživih nedeljivih resursa. Kapacitet rezervoara je `PoolSize` identičnih instanci istog tipa resursa. Sve instance su inicijalno slobodne. Proces traži zauzeće jedne instance resursa pozivom funkcije `request` monitora. Ukoliko slobodnih instanci resursa nema, proces mora da čeka dok neka instanca resursa ne postane slobodna. Instanca resursa koja je dodeljena procesu identifikuje se celim brojem u opsegu `0..PoolSize-1` i vraća se kao rezultat funkcije `request`. Kada završi sa korišćenjem resursa, proces poziva operaciju `release` monitora koja svojim parametrom zadaje identifikaciju instance resursa koja se oslobađa.

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Implementirati veb servis na programskom jeziku Java, koji uparuje korisnike. Korisnik se prilikom uspostavljanja veze sa serverom loguje, pa šalje serveru IP adresu i port na kome će uspostaviti vezu sa uparenim korisnikom. Logovanje nije potrebno realizovati, već samo staviti komentar gde bi taj kod trebao da stoji. Logovanje može potrajati, pa svu komunikaciju sa jednim klijentom treba obraditi u posebnoj niti. Server uparuje dva uzastopna korisnika koji mu pošalju IP adresu i port i to tako sto drugom korisniku naloži da se javi prvom na IP adresu i port koji je prvi korisnik dostavio prilikom prijave na server. Korisnici su jednonitni procesi. Dozvoljeno je korišćenje koda prikazanog na vežbama (kod sa vežbi ne treba prepisivati, nego npr. reći koja klasa ili koji metod se koriste i/ili menjaju, nasleđuju, ...). Napisati samo kod servera kao rešenje zadatka. Server prihvata zahteve klijenata na portu 5555.

Rešenje: