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 2019.
Datum: 14. 1. 2019.

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 primenom algoritma *round-robin*, ali uz tzv. *grupno raspoređivanje* (engl. *group scheduling*) na sledeći način. Spremni procesi su grupisani u *grupe* (proces se dodeljuje nekoj grupi prilikom kreiranja). Svakoj grupi dodeljen je ukupan maksimalan vremenski interval izvršavanja, a procesi u datoj grupi dele taj ukupan interval na približno jednake intervale svog izvršavanja. Na taj način se postiže željena raspodela procesorskog vremena između grupa procesa; na primer, ako je procesima korisnika *A*, kao jednoj grupi procesa, dodeljeno ukupno 50% procesorskog vremena, a procesima korisnika *B* kao drugoj grupi procesa preostalih 50%, i ako korisnik *A* ima 5 svojih kreiranih procesa, a korisnik *B* 100 svojih procesa, onda će svaki proces korisnika *A* dobiti najviše po 10% (približno), a korisnika *B* po 0,5% procesorskog vremena (naravno, proces se može izvršavati i kraće, ako se suspenduje pre isteka dodeljenog intervala). Tako procesi iz jedne grupe ne mogu da ugrožavaju izvršavanje procesa iz druge grupe.

Struktura spremnih procesa organizovana je na sledeći način. Klasa `Scheduler` sadrži dvostruko ulančanu listu grupa (grupa je predstavljena klasom `ProcGroup`). Svaka grupa sadrži (potencijalno praznu) listu spremnih procesa te grupe (proces je predstavljen klasom `PCB`). Pokazivači `next` i `prev` su pokazivači za ulančavanje u ove liste, a `head` i `tail` glava i rep tih lista. Polje `size` klase `ProcGroup` sadrži trenutni broj procesa u datoj grupi, a polje `slice` ukupan vremenski interval koji treba dati celoj grupi (najviše).

Operacija `Scheduler::get` treba da vrati proces koji sledeći treba da se izvršava i da u njegovo polje `PCB::slice` upiše vremenski interval koji tom procesu treba dodeliti za izvršavanje. Ova operacija *ne treba* da izbacuje taj proces iz liste spremnih: to će uraditi ostatak kernela samo ako se proces suspenduje, pozivom operacije `Scheduler::remove`. Drugim rečima, proces ostaje na istom mestu u listih spremnih sve dok se ne suspenduje i izbaci operacijom `remove`. Tip `Time` je celobrojni tip velikog opsega.

Pretpostavlja se da skup spremnih procesa ima najmanje jednu grupu i najmanje jedan proces, ali ne obavezno u svakoj grupi (mogu postojati prazne grupe, ali je uvek bar jedna neprazna).

Realizovati operaciju `Scheduler::get`.

```
struct PCB { PCB *next, *prev;  Time slice; ... };

struct ProcGroup {
  ProcGroup *next, *prev;
  PCB *head, *tail;
  unsigned size;
  Time slice;
};

class Scheduler {
public:
  Scheduler () {...}
  PCB* get ();
  void remove (PCB*);
  void put (PCB*);
private:
  ProcGroup *head, *tail;
  ...
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Na jeziku Java napisati kod za monitor koji ima dve procedure, *tick* i *tuck*, i prihvata pozive na njima na sledeći način: neki proces može izvršiti najpre proceduru *tick*, i to najmanje jednom (jednom ili više puta), a nakon toga neki proces može izvršiti proceduru *tuck* tačno jednom, nakon čega može ponovo *tick* jednom ili više puta i tako ciklično (npr. *tick-tick-tuck-tick-tuck-tick-tick-tick-tuck...*).

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Na programskom jeziku Java napisati kod serverskog programa koji prima poruke i prosleđuje ih klijentima. Server preko priključnice, koja mu se prosleđuje prilikom kreiranja, prima poruke. Poruka se sastoji iz dve linije teksta. Prva linija predstavlja ključ koji jedinstveno određuje primaoca, dok je druga linija sadržaj poruke. Server osluškuje na portu 5555, preko koga mu se javljaju klijenti. Svaki klijent pošalje serveru svoj ključ kojim se identifikuje. Server klijentu šalje sadržaj poruke koja je namenjena tom klijentu. Za svakog klijenta biće poslata tačno jedna poruka. Server može uporedo da opslužuje više klijenata. Dozvoljeno je korišćenje koda datog na vežbama.

Rešenje: