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 2022.
Datum: 20. 1. 2022.

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`, čiji je interfejs dat dole, realizuje raspoređivač spremnih procesa. Raspoređivanje je po prioritetu. Svaki proces ima svoj prioritet u opsegu `0..MaxPri` (niža vrednost označava viši prioritet). U skupu procesa istog prioriteta raspoređivanje je *round-robin*.

Procesi tokom svog izvršavanja ne menjaju prioritet koji im je inicijalno dodeljen. Zbog toga raspoređivač održava `MaxPri+1` kružno ulančanih lista procesa, po jednu za svaki od mogućih prioriteta. Prilikom kreiranja procesa, ostatak kernela dodaje proces u odgovarajuću listu pozivom operacije `add`; slično, prilikom gašenja procesa, ostatak kernela poziva operaciju `remove` da bi proces izbacio iz njegove liste.

Prilikom izbora procesa za izvršavanje pozivom operacije `get`, proces se ne izbacuje iz ove svoje liste u raspoređivaču, već tu ostaje do svog gašenja. Da je proces spreman za izvršavanje ukazuje polje `isRunnable` u strukturi PCB. U strukturi PCB dostupna su polja `priority` (tekući prioritet procesa) i `next` za ulančavanje u listu.

Implementirati u potpunosti klasu `Scheduler` .

```
struct PCB { PCB *next;  Priority priority; bool isRunnable; ... };
const unsigned MaxPri = ...;

class Scheduler {
public:
  Scheduler ();
  void add (PCB*);
  void remove (PCB*);

  PCB* get ();
};
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Korišćenjem klasičnih uslovnih promenljivih realizovati monitor koji se koristi za kontrolu ekskluzivnog pristupa deljenom resursu. Kada proces želi da koristi resurs, mora najpre da pozove operaciju `acquire` ovog monitora. Kada proces oslobađa resurs, mora da pozove operaciju `release` ovog monitora. Ukoliko je resurs zauzet, proces mora da čeka, ali tako da najviše `MaxWaiting` procesa može čekati na resurs. Ukoliko proces zatraži resurs na kom već čeka maksimalan broj procesa, operacija `acquire` odmah vraća kontrolu sa rezultatom `false`. Kada je proces dobio pristup resurus, operacija `acquire` vraća kontrolu sa rezultatom `true`.

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Napisati kod servera na programskom jeziku Java koji opslužuje zahteve klijenata na portovima od 5555 do 6666. Server osluškuje na nekom portu (na početku na portu 5555). Ukoliko se u toku trideset sekundi ne pojavi nijedan klijent, server treba da prekine osluškivanje na tom portu i treba da počne da osluškuje na portu čiji je broj za jedan veći. Kada se dođe do porta 6666 treba opet krenuti osluškivanje na portu 5555. Prekidanje osluškivanja se radi zatvaranjem serverske priključnice (poziv metode `close` objekta `ServerSocket`). Komunikacija sa klijentom se radi pomoću date klase `RequestHandler` čiji je interfejs:

```
public class RequestHandler extends Thread {
    public static class Service {
        public Service(Socket socket);
    }
    public RequestHandler(Service service, Server server);
    public void run();
}
```

Rešenje: