Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo
Kolokvijum: Prvi, decembar 2025.
Datum: 1. 12. 2025.

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

Klasa `Scheduler` implementira raspoređivanje po prioritetima, a po *Round-Robin* algoritmu za procese istog prioriteta. Raspoređivanje takođe podržava grupnu raspodelu vremena kojom osigurava garantovan „vremenski budžet“ raspoloživ za izvršavanje svakoj grupi procesa istog prioriteta (*group scheduling*). Vremenski budžet svake grupe procesa dopunjava se svake vremenske periode do vremena definisanih u nizu `timeBudgets` pozivom operacije `refillTime`. Trenutno ukupno preostalo vreme za izvršavanje procesa datog prioriteta zapisano je u elementu niza `times`. Procesu koji je izabran za izvršavanje treba dodeliti ograničeno vreme izvršavanja (`PCB::setTimeSlice`) koje mu je podrazumevano (`PCB::getDefaultTimeSlice`), ali ne duže od vremena koje je preostalo za izvršavanje njegovoj grupi. Kada odabrani proces izgubi procesor, kernel poziva operaciju `Scheduler::updateTime` dajući informaciju o stvarno izmerenom proteklom vremenu izvršavanja tog procesa datog prioriteta. Tip `Time` je dovoljno velik tip neoznačenih celih brojeva i predstavlja vremenski interval.

Implementirati operacije `get` i `updateTime` klase `Scheduler`. Na raspolaganju je i klasa `Queue` koja implementira FIFO red sa operacijama `get` i `put`.

```
class PCB {
public:
  Time getDefaultTimeSlice () const;
  void setTimeSlice (Time slice);
  ...
};

class Scheduler {
public:
  Scheduler () {}
  PCB* get ();
  void updateTime (unsigned priority, Time execTime);

private:
  Time times[MAXPRI];
  static const Time timeBudgets[MAXPRI];
  Queue queues[MAXPRI];
};

PCB* Scheduler::refillTime () {
  for (unsigned i=0; i<MAXPRI; i++)
    times[i] = timeBudgets[i];
}
```

Rešenje:

2. (10 poena) Međuprocesna komunikacija pomoću deljene promenljive

Korišćenjem brojačkih semafora implementirati klasu `ReaderWriters` koja obezbeđuje potrebnu sinhronizaciju pristupa deljenom podatku tipa `T` od strane jednog čitaoaca i više pisaca na sledeći način:

- postoje dve instance tipa `T` za deljeni pristup, nalaze se u elementima niza `data[2]`;
- u jednom periodu jedna od te dve instance služi za čitanje, a druga za upis;
- operacijom `swap` ove dve instance zamenjuju uloge;
- operacija `read` vraća pokazivač na instancu koja je trenutno namenjena za čitanje, ali samo ako je ona pre toga barem jednom upisana nakon inicijalizacije ili je bila pozvana operacija `swap`, inače vraća *null*;
- operacija `write` upisuje u instancu koja je trenutno namenjena za upis; tip `T` može se kopirati operatorom dodele i nije skalaran.

```
class ReaderWriters {
public:
  ReaderWriters ();
  const T* read () const;
  void write (const T&);
  void swap ();

private:
  T data[2];
  ...
};
```

Rešenje:

3. (10 poena) Međuprocesna komunikacija razmenom poruka

Napisati kod servera koji omogućava korišćenje hardverskih uređaja. Na klijentskoj strani interfejs za korišćenje uređaja je:

```
public class Devices {
  public List<String> executeOnDevice(String deviceId,
                                       List<String> commands);
}
```

Stringovi koji predstavljaju komande i rezultat mogu sadržati bilo koji karakter osim prelaska u novi red.

Server ima metodu:

```
public List<String> executeOnDevice(String deviceId, List<String> commands);
```

koju nije potrebno implementirati i koja nije bezbedna za rad u višenitnom okruženju. Server poseduje i listu svih dostupnih uređaja:

```
private List<String> deviceIds.
```

Uređaji su nedeljivi i operacije nad njima moraju biti atomične. Server treba da obrađuje zahteve više klijenata uporedo. Server osluškuje na portu 5555.

Rešenje: