Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Treći, jun 2026.
Datum: 5. 7. 2026.

Treći kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. Dozvoljeno je korišćenje literature.

Zadatak 1 _____/10 *Zadatak 3* _____/10
Zadatak 2 _____/10

Ukupno: _____/30 = _____% = _____/15

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) Ulaz/izlaz

Date su deklaracije pokazivača preko kojih se može pristupiti registrima kontrolera tastature preko koga stižu znakovi otkucani na tastaturi:

```
typedef volatile unsigned REG;
REG* ioStatus =...;    // status register
volatile char* ioData =...;     // data register
```

Kontroler tastature poseduje interni memorijski modul koji služi za prihvatanje jednog ili više znakova otkucanih na tastaturi (hardverski bafer). Kada se u ovom baferu pojave znakovi (jedan ili više u istom naletu), kontroler generiše prekid. Tada se iz registra za podatke mogu čitati pristigli znakovi sukcesivnim operacijama čitanja bez čekanja, jedan po jedan, sve dok je bit spremnosti (*ready*) u razredu 0 statusnog registra postavljen na 1. Naredni nalet pristiglih znakova će ponovo generisati prekid.

Za smeštanje znakova učitanih sa tastature, svaki proces poseduje i koristi svoj (softverski) ograničeni bafer veličine 256 znakova koji mu je dodelio kernel. Kada u korisničkom interfejsu korisnik premesti fokus na prozor koji pripada nekom procesu, kernel postavlja tekući bafer tastature na bafer tog procesa pozivom operacije `KeyboardBuffer::setCurrent`; ukoliko takvog procesa nema, kernel postavlja tekući bafer na *null*. U ovaj tekući bafer upisuju se znakovi učitani sa kontrolera tastature sve dok u baferu ima mesta; znakovi koji ne mogu da stanu u bafer se jednostavno odbacuju, a isto se dešava i ako je tekući bafer postavljen na *null*. Iz ovog bafera znakove uzima proces kome pripada dati bafer pozivom operacije `KeyboardBuffer::getc`; ukoliko u baferu nema znakova, pozivajući proces treba da se suspenduje dok znakova ne bude. Operacija `KeyboardBuffer::putc` je namenjena za korišćenje iz prekidne rutine i treba da smesti jedan znak u tekući bafer. Sinhronizacija se može obavljati semaforima čiji je interfejs isti kao u školskom jezgru. Pretpostaviti sledeće:

- prekidna rutina izvršava se međusobno isključivo sa operacijama na semaforu;
- procesor maskira prekide pri obradi prekida; prekidi nisu podrazumevano maskirani u ostatku operacija bafera tastature; prekid sa kontrolera tastature se može eksplicitno maskirati pozivom `kbint_mask()`, a demaskirati pozivom `kbint_unmask()`;
- operacija `signal` na semaforu može se pozivati iz prekidne rutine, jer ona ne radi nikakvu promenu konteksta.

Implementirati opisani podsistem: bafer procesa (operacije `getc`, `putc` i sve druge potrebne operacije klase `KeyboardBuffer` čiji je interfejs dat) i prekidnu rutinu kontrolera tastature.

```
class KeyboardBuffer {
public:
  KeyboardBuffer();
  char getc ();
  void putc (char c);

  static KeyboardBuffer* getCurrent ();
  static void setCurrent (KeyboardBuffer* kb);
};

interrupt void keyb_int ();
```

Rešenje:

2. (10 poena) Fajl sistem

Korišćenjem sistemskog poziva *popen* (*pipe open*), na jeziku C napisati program *redirect* koji se iz komandne linije poziva ovako:

`redirect` *`program input_file`*

Ovim se pokreće proces nad programom *program*, koji radi tako što znakove učitava sa svog standardnog ulaza, tako što mu se ti znakovi obezbeđuju iz datog ulaznog tekstualnog fajla *input_file*. Ignorisati sve greške. Dati su potpisi odgovarajućih raspoloživih sistemskih poziva iz *libc stdio stream* API (`<stdio.h>`); Funkcija `fgetc` vraća `EOF` ako je učitavanje stiglo do kraja fajla:

```
FILE* popen (const char* filename, const* char mode);
int   pclose (FILE*);
FILE* fopen (const char* filename, const* char mode);
int   fclose (FILE*);
int   fgetc (FILE*);
int   fputc (int ch, FILE*);
```

Rešenje:

3. (10 poena) Fajl sistem

Neki fajl sistem implementira direktorijume kao i fajlove, zapisujući sadržaj direktorijuma kao sadržaj fajla. Preslikavanje simboličkog imena u ulaz u direktorijumu implementirano je pomoću heš tabele na sledeći način. U prvom bloku sadržaja direktorijuma nalazi se objekat klase `Dir` koja je data dole; u njemu je heš tabela koja sadrži glave lista ulaza u direktorijumu koji se preslikavaju u isti ulaz heš tabele. U narednim blokovima sadržaja direktorijuma zapisane su te ulančane liste; svaki element liste je jedan ulaz u direktorijumu predstavljen objektom klase `DirEntry`. Svi ti elementi su složeni jedan iza drugog i numerisani kao „slotovi“ rednim brojevima; ulaz u heš tabeli sadrži broj slota u kom je prvi zapis u listi, dok svaki zapis (objekat `DirEntry`) sadrži broj slota sledećeg zapisa u istoj listi. Date su implementirane funkcije koje rade sledeće:

- `Dir::hash`: heš funkcija koja za dato simboličko ime vraća broj ulaza u heš tabeli u koje se to ime preslikava, u opsegu 0..`HASH_TABLE_SIZE-1`;
- `Dir::getDentry`: za dati broj slota vraća pokazivač na objekat tipa `DirEntry` koji se nalazi u tom slotu, po potrebi učitavajući odgovarajući blok u kom je taj objekat; ukoliko data vrednost za broj slota označava kraj liste, ova funkcija vraća *null*;
- `DirEntry::getFCBBlockNo`: vraća broj bloka u kom se nalazi FCB datog ulaza u direktorijumu;
- `DirEntry::getNext`: vraća broj slota u kom je sledeći zapis u istoj listi;
- `DirEntry::cmpName`: poredi dato simboličko ime sa imenom u ovom ulazu i vraća `true` ako su ta imena ista.

Implementirati funkciju `Dir::getFCBBlock` koja treba da pronađe dato simboličko ime u direktorijumu čiji je prvi blok učitan u memoriju (to je objekat klase `Dir` na koga ukazuje `this`) i vrati broj bloka u kom je FCB u koji se to simboličko ime preslikava. Ako datog imena u direktorijumu nema, ova funkcija treba da vrati 0.

```
class Dir {
public:
  size_t    hash (const char name[FNAME_LEN]) const;
  DirEntry* getDentry (size_t slot) const;

  BlockNo   getFCBBlock (const char name[FNAME_LEN]) const;
private:
  size_t hashTbl[HASH_TABLE_SIZE];
};

class DirEntry {
public:
  BlockNo getFCBBlockNo () const { return blk; }
  size_t  getNext() const { return nxt; }
  bool cmpName (const char name[FNAME_LEN]) const;
private:
  char fname[FNAME_LEN]; BlockNo blk; size_t nxt;
};
```

Rešenje: