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 2022.
*Datum:* 12. 6. 2022.

# *Treći kolokvijum iz Operativnih sistema 1*

*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 = _____% = _____/10

**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
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 ovom baferu pojave znakovi (jedan ili više), kontroler generiše prekid. Tada se iz registra za podatke mogu čitati pristigli znakovi sukcesivnim operacijama čitanja, 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 kernel koristi svoj (softverski) ograničeni bafer veličine 256 znakova. U ovaj bafer upisuju se znakovi učitani sa kontrolera tastature sve dok u njemu ima mesta; znakovi koji ne mogu da stanu u bafer se jednostavno odbacuju. Iz ovog bafera znakove uzimaju različiti uporedni tokovi kontrole (procesi) pozivom operacije `getc`; ukoliko u baferu nema znakova, pozivajući tok kontrole treba da se suspenduje dok znakova ne bude. 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;
- operacija `signal` na semaforu može se pozivati iz prekidne rutine, jer ona ne radi nikakvu promenu konteksta;
- procesor maskira prekide pri obradi prekida; prekid sa kontrolera tastature se može eksplicitno maskirati pozivom `kbint_mask()`, a demaskirati pozivom `kbint_unmask()`.

Implementirati opisani podsistem: bafer kernela (operaciju `getc` i sve druge potrebne operacije) i prekidnu rutinu kontrolera tastature.

Rešenje:

## 2. (10 poena) Fajl sistem

Neprazno binarno stablo čiji su čvorovi tipa `Node` zapisano je u binarni fajl na sledeći način (prikazan je pojednostavljen kod bez obrade grešaka): za svaki čvor najpre je zapisan njegov sadržaj tipa `NodeData`, zatim jedan `int` indikator koji kaže da li taj čvor ima svoje levo podstablo, zatim jedan `int` indikator koji kaže da li taj čvor ima svoje desno podstablo, a onda isto tako redom celo levo podstablo ako ga ima, pa celo desno podstablo ako ga ima. Korišćenjem istog POSIX API za fajlove napisati kod funkcije `readTree` za učitavanje i izgradnju stabla iz takvog fajla. Prvi parametar je staza do fajla, a drugi parametar je adresa pokazivača u koji treba upisati adresu korenog čvora formiranog i učitanog stabla. Obraditi greške vraćanjem negativne vrednosti. POSIX API funkcija

```
ssize_t read(int fd, void *buf, size_t count);
```

vraća broj stvarno učitanih bajtova (može biti manji od zahtevanog ako se stigne do kraja fajla) ili negativnu vrednost u slučaju greške.

```
#include <fcntl.h>

struct Node {
  NodeData data;
  Node *left, *right;
};

void writeSubtree (int fd, Node* node) {
  write(fd,node->data,sizeof(NodeData));
  int ind = (node->left!=0);
  write(fd,ind,sizeof(int));
  ind = (node->right!=0);
  write(fd,ind,sizeof(int));

  if (node->left) writeSubtree(fd, node->left);
  if (node->right) writeSubtree(fd, node->right);
}

void writeTree (const char* pathname, Node* root) {
 int fd = open(*pathname,O_WRONLY|O_CREATE|O_TRUNC);
 int r = writeSubtree(fd,root);
 close(fd);
}

int readTree (const char* pathname, Node** root);
```

Rešenje:

## 3. (10 poena) Fajl sistem

Sistemski pozivi za pristup sadržaju binarnog fajla uobičajeno omogućavaju čitanje ili upis *size* susednih bajtova počev od pozicije *offset* u odnosu na početak sadržaja fajla (prvi bajt je na poziciji 0). Da bi ostvario ovakav pristup, neki kernel koristi klasu `FLogicalAccess` čiji je interfejs dat dole. Ova klasa koristi se tako što se za svaki ovakav pristup instancira objekat ove klase, inicijalizuje se pozivom operacije `reset` sa zadatim parametrima, a onda se iterira sve dok operacija `end` ne vrati 1. U svakoj iteraciji se pristupa odgovarajućim bajtovima jednog logičkog bloka fajla, tako što se može dobiti sledeća informacija:

- `getBlock`: vraća redni broj logičkog bloka u kom se pristupa u toj iteraciji;
- `getRelOffset`: vraća redni broj bajta u tekućem bloku počev od kog se pristupa;
- `getRelSize`: broj bajtova u tekućem bloku kojima se pristupa u toj iteraciji.

```
class FLogicalAccess {
public:

 FLogicalAccess ();
 void reset (size_t offset, size_t size);
 int  end() const;
 void next();

 size_t getBlock() const;
 size_t getRelOffset() const;
 size_t getRelSize() const;
};
```

Primer upotrebe ove klase je sledeći:

```
FLogicalAccess fla;
for (fla.reset(offset,size); !fla.end(); fla.next()) {
  // Access fla.getRelSize() bytes
  // starting from the offset fla.getRelOffset()
  // in the logical block fla.getBlock()
}
```

Na primer, ako je veličina bloka 8 bajtova, a inicijalno je zadato: *offset* = 11 i *size* = 15, onda će u prvoj iteraciji biti: `getBlock()`=1, `getRelOffset()`=3, `getRelSize()`=5, u drugoj će ove vrednosti redom biti 2, 0, 8, a u trećoj 3, 0, 2; posle toga će se izaći iz petlje zbog `end()`=1.

Implementirati u potpunosti klasu `FLogicalAccess`. Veličina bloka je `BLK_SIZE`.

Rešenje: