Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 1 (SI2OS1, IR2OS1)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Treći, jun 2014.
Datum: 20.6.2014.

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

U nekom sistemu svaki drajver blokovski orijentisanog uređaja („diska“) registruje sledeću strukturu (tabelu) koja sadrži pokazivače na funkcije koje implementiraju odgovarajuće operacije sa tim uređajem:

```
typedef ... Byte; // Unit of memory
typedef ... BlkNo; // Disk block number

struct DiskOperationsTable {
  int (*readBlock)(BlkNo block, Byte* buffer);
  int (*writeBlock)(BlkNo block, Byte* buffer);
};
```

Sistem organizuje tabelu registrovanih drajvera za priključene uređaje kao niz pokazivača na ove strukture:

```
const int MaxNumOfDisks; // Maximal number of registered disk devices

DiskOperationsTable* disks[MaxNumOfDisks];
```

Sistem preslikava simbolička imena dodeljena priključenim uređajima, u obliku slova abecede, brojevima ulaza u tabeli `disks` (u opsegu od 0 do `MaxNumOfDisks-1`). Ukoliko uređaj sa datim simboličkim imenom nije priključen, odgovarajući ulaz u ovoj tabeli je *null*.

Realizovati funkcije:

```
int readBlock(int diskNo, BlkNo block, Byte* buffer);
int writeBlock(int diskNo, BlkNo block, Byte* buffer);
```

koje treba da pozovu odgovarajuću implementaciju operacije drajvera (polimorfno, dinamičkim vezivanjem) za zadati uređaj. (Ove funkcije poziva interna kernel nit kada opslužuje zahteve za operacijama sa diskovima, da bi inicijalizovala prenos na odgovarajućem uređaju.) U slučaju greške, sve ovde navedene funkcije vraćaju negativnu vrednost, a u slučaju uspeha vraćaju 0.

Rešenje:

2. (10 poena) Fajl sistem

U nekom fajl sistemu primenjuje se zaštita pristupa fajlovima kao u sistemu Unix. U strukturi FCB postoje sledeća polja:

- `unsigned long int owner`: identifikator korisnika koji je vlasnik fajla;
- `unsigned long int group`: identifikator grupe korisnika kojoj je fajl pridružen;
- `unsigned int protection`: biti prava pristupa (relevantno je samo 9 najnižih bita, pri čemu najviša 3 su dodeljena vlasniku, naredna 3 grupi, a najniža 3 ostalima).

U strukturi UCB *(user control block)* koja predstavlja jednog registrovanog korisnika sistema takođe postoji polje `group` koje predstavlja identifikator grupe kojoj je taj korisnik pridružen. Prava pristupa za grupu kojoj je pridružen fajl odnose se na korisnike koji su pridruženi istoj toj grupi.

Na raspolaganju je i sledeća funkcija koja vraća pokazivač na odgovarajuću strukturu UCB za korisnika koji je identifikovan datim identifikatorom:

```
UCB* getUCB (unsigned long int uid);
```

Realizovati funkciju koja ispituje da li je tražena operacija dozvoljena datom korisniku za dati fajl. Operacija se identifikuje jednim od tri najniža bita *rwx*, sa značenjem kao u bitima prava pristupa za fajl, s tim što je uvek samo jedan bit postavljen na 1. Funkcija treba da vrati 1 ako je operacija dozvoljena, a 0 ako nije.

```
int isAllowed(FCB* file, unsigned long int uid, unsigned int op);
```

Rešenje:

3. (10 poena) Fajl sistem

U implementaciji nekog fajl sistema evidencija slobodnih blokova na disku vodi se na sledeći način. Indeks (spisak) slobodnih blokova je neograničen i zapisuje se u samim slobodnim blokovima, ulančanim u jednostruku listu. Prema tome, prvi slobodan blok sadrži spisak najviše N slobodnih blokova (*ne* uključujući njega samog, tj. on nije na spisku), dok poslednji ulaz u tom spisku u tom bloku sadrži broj sledećeg bloka u listi, u kome se nalazi nastavak spiska slobodnih blokova itd. Ukoliko je neki slobodni blok bio na spisku, a više nije, njegov ulaz u indeksu slobodnih blokova ima vrednost 0 (blok broj 0 nikada nije slobodan). Kada se zahteva jedan slobodan blok, treba jednostavno uzeti prvi slobodan blok sa spiska. Pri tome, ukoliko je ceo spisak sadržan u prvom bloku u listi ispražnjen, treba alocirati upravo taj prvi blok iz liste i izbaciti ga iz liste.

Na prvi blok u listi ukazuje globalna promenljiva `freeBlocksHead`. Na raspolaganju je i funkcija `getBlock` koja vraća pokazivač na deo memorije u kešu u koju je učitan traženi blok sa diska. (Napomena: broj N je određen veličinom bloka i veličinom tipa `BlockNo`.)

```
typedef … BlockNo; // Disk block number
extern int blockSize; // Disk block size
void* getBlock (BlockNo block);
extern BlockNo freeBlocksHead;
```

Realizovati funkciju `getFreeBlock()` koja treba da alocira i vrati jedan slobodni blok:

```
BlockNo getFreeBlock ();
```

Rešenje: