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 2016.
Datum: 19.6.2016.

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 izvršena je adaptacija interfejsa znakovno orijentisanog sekvencijalnog izlaznog uređaja na blokovski orijentisani sekvencijalni uređaj pomoću ograničenog bafera. Proizvođači, koji su uporedne niti, upisuju u bafer sekvencijalno bajt po bajt, operacijom `put`. Potrošači, takođe uporedne niti, uzimaju po ceo blok podataka veličine `BlockSize` iz bafera operacijom `read` i prenose ih dalje na izlazni uređaj.

Implementirati klasu `Buffer` čiji je interfejs dat dole. Sinhronizaciju vršiti pomoću semafora školskog jezgra, koji su, radi pogodnosti upotrebe za ovu namenu, prošireni operacijom `signal(unsigned)` koja atomično inkrementira vrednost semafora za vrednost zadatog argumenta (koji može biti proizvoljan prirodan broj, a ne samo jedan, kako je podrazumevano).

```
typedef unsigned short Byte;
const int BlockSize = ...;
const int NumOfBlocks = ...;
const int BufferSize = NumOfBlocks*BlockSize;

class Buffer {
public:
  Buffer ();
  void put (Byte b);
  void read (Byte block[]);
};
```

Rešenje:

2. (10 poena) Fajl sistem

U implementaciji nekog fajl sistema svaki čvor u hijerarhijskoj strukturi direktorijuma i fajlova predstavljen je objektom klase `Node`. Operacija te klase:

```
Node* Node::getSubnode(const char* pStart, const char* pEnd);
```

vraća podčvor datog čvora `this` koji ima simboličko ime zadato nizom znakova koji počinje znakom na koga ukazuje `pStart`, a završava znakom ispred znaka na koga ukazuje `pEnd` (`pEnd` može ukazivati na `'\0'` ili `'/'`). Ukoliko dati čvor `this` nije direktorijum, ili u njemu ne postoji podčvor sa datim simboličkim imenom, ova funkcija vraća 0.

Za svaki proces se u polju `curDir` strukture PCB čuva pokazivač na čvor (tipa `Node*`) koji predstavlja tekući direktorijum datog procesa. Koreni direktorijum cele hijerarhije dostupan je kao statički pokazivač `Node::rootNode` tipa `Node*`.

Znak za razdvajanje (delimiter) u stazama, kao i znak za koreni direktorijum je kosa crta `'/'`. Implementirati sledeću funkciju koja se koristi u implementaciji ovog fajl sistema:

```
Node* Node::getNode (PCB* pcb, const char* path);
```

Ova funkcija vraća čvor određen stazom koja je zadataka drugim argumentom, pri čemu ta staza može biti zadata kao apsolutna (počinje znakom `'/'`), ili kao relativna (ne počinje znakom `'/'`) u odnosu na tekući direktorijum procesa čiji je PCB dat kao prvi argument.

Rešenje:

3. (10 poena) Fajl sistem

U implementaciji nekog FAT fajl sistema cela FAT keširana je u memoriji u strukturi:

```
extern unsigned long fat[];
unsigned long freeHead;
```

Za ulančavanje se kao *null* vrednost u ulazu u FAT koristi 0 (blok broj 0 je rezervisan). U FCB fajla polje `head` sadrži redni broj prvog bloka sa sadržajem fajla. Slobodni blokovi su takođe ulančani u FAT, a broj prvog slobodnog bloka u lancu sadrži globalna promenljiva `freeHead`.

Implementirati sledeću funkciju:

```
unsigned long append (FCB* fcb);
```

koja treba da proširi sadržaj fajla sa datim FCB za jedan blok (da doda jedan blok na kraj). Funkcija vraća 0 ako slobodnih blokova nema ili u slučaju bilo koje druge greške, odnosno broj alociranog bloka ako je operacija uspela. Ne treba vršiti nikakve optimizacije u smislu alokacije najbližeg bloka već alociranim blokovima datog fajla.

Rešenje: