Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Operativni sistemi 2 (SI3OS2, IR3OS2)
Nastavnik: prof. dr Dragan Milićev
Odsek: Softversko inženjerstvo, Računarska tehnika i informatika
Kolokvijum: Treći, januar 2017.
Datum: 17.1.2017.

Treći kolokvijum iz Operativnih sistema 2

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 = _____%

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) Upravljanje diskovima

U nekom sistemu klasa `DiskScheduler` data dole realizuje raspoređivač zahteva za pristup disku po *C-SCAN* algoritmu, ali uz sprečavanje izgladnjivanja ograničavanjem vremena čekanja zahteva na opsluživanje, na sličan način kako to radi i Linux (prilikom smeštanja novog zahteva u red, zahtevu se dodeljuje krajnji vremenski rok, engl. *deadline*, do kog se zahtev mora opslužiti). Prilikom smeštanja zahteva tipa `DiskRequest` u red zahteva operacijom `put()`, zahtev se smešta u dve dvostruko ulančane liste: prvu uređenu po broju cilindra na koji se zahtev odnosi, za *C-SCAN* algoritam (kružna lista u kojoj `scanHead` ukazuje na zahtev koji je naredni na redu za opsluživanje), i drugu uređenu hronološki, po vremenu isteka roka do kog se zahtev mora opslužiti (lista čija je glava `edfHead`, EDF – *Earliest Deadline First*). U strukturi `DiskRequest` polja `edfNext` i `edfPrev` predstavljaju pokazivače za (dvostruko) ulančavanje u *EDF* listu, a polja `scanNext` i `scanPrev` pokazivače za (dvostruko) ulančavanje u *C-SCAN* listu; polje `deadline` sadrži razliku između trenutka isteka vremenkog roka datog zahteva i trenutka isteka vremenskog roka zahteva koji mu hronološki prethodi u listi EDF; za prvi zahtev u toj listi, ovo polje predstavlja relativno vreme isteka vremenskog roka tog zahteva u odnosu na sadašnji trenutak (može biti i negativno ako je taj rok istekao).

Implementirati operaciju `get` klase `DiskScheduler` koja iz reda uzima zahtev koji sledeći treba opslužiti.

```
struct DiskRequest {
  DiskRequest *edfNext, *edfPrev;
  DiskRequest *scanNext, *scanPrev;
  long deadline;
  ...
};

class DiskScheduler {
public:
  DiskScheduler ();
  DiskRequest* get ();
  void put (DiskRequest*);
private:
  DiskRequest *edfHead, *scanHead;
};

DiskScheduler::DiskScheduler () : edfHead(0), scanHead(0) {}
```

Rešenje:

2. (10 poena) Operativni sistem Linux

Napisati *Bash* skriptu koja dodaje prava pristupa fajlovima čije ime sadrži određeni šablon. Šablon je zadat kao prvi argument u vidu regularnog izraza. Drugi argument skripte je ime direktorijuma u kome treba menjati prava pristupa fajlova. Treći argument je jedno slovo koje označava koje pravo pristupa treba dodati fajlovima (*r*, *w* ili *x*). Pravo pristupa se dodaje za sve korisnike i dodaje se samo pravim fajlovima (ne direktorijumima). Fajl kome se menja pravo pristupa može biti na proizvoljnoj dubini u hijerarhiji direktorijuma koji je zadat kao drugi argument. U slučaju neispravnog broja argumenata prijaviti grešku i prekinuti izvršavanje skripte.

Rešenje:

3. (10 poena) Operativni sistem Linux

Napisati program na programskom jeziku C/C++ koji napravi NUM procesa koji međusobno komuniciraju (početni proces se uračunava među NUM procesa). NUM je pozitivna konstanta. Svaki proces treba da ima redni broj koji označava redosled po kome su procesi kreirani (redni broj početnog procesa je 1). Svaki proces na početku dobija neku realnu vrednost koja je jednaka njegovom rednom broju. Procesi obavljaju posao u koracima. Jedan korak se sastoji od sledećeg posla: proces podeli svoju vrednost na dva dela i te vrednosti pošalje susednim procesima, zatim od susednih procesa primi dve vrednosti koje sabere i taj zbir postaje njegova nova vrednost. Broj koraka je definisan kao pozitivna konstanta. Susedni procesi su oni koji imaju za jedan veći i za jedan manji redni broj od posmatranog procesa. Procesi sa prvim i poslednjim rednim brojem su međusobno susedni. Odnos primljenih vrednosti u prethodnom koraku definiše odnos prema kojem će biti podeljena vrednost u tekućem koraku. U prvom koraku se vrednost deli na pola. Za međuprocesnu komunikaciju koristiti neimenovane cevi operativnog sistema Linux.