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 2015.
Datum: 13.1.2015.

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) Sistemski pozivi i arhitektura operativnih sistema

Neki operativni sistem ima mikrokernel arhitekturu. U kernelu ovog sistema implementirane su samo osnovne funkcije, uključujući upravljanje procesima i međuprocesnu komunikaciju, ali ne i fajl podsistem. Na raspolaganju su sledeći sistemski pozivi:

```
int getPID();  // Returns the current process ID
int send (int receiver_process_ID, const char* message, size_t msg_size);
int receive (int sender_process_ID, char* buffer, size_t buffer_size);
```

Prvi od ovih poziva vraća identifikator pozivajućeg procesa. Ostala dva poziva služe za međuprocesnu komunikaciju razmenom poruka. Komunikacija je sa direktnim i simetričnim imenovanjem, pri čemu se proces-sagovornik identifikuje njegovim identifikatorom. Poruka je proizvoljan sadržaj zadate dužine. Kod prijema poruke u sistemskom pozivu `receive` drugim i trećim argumentom zadaju se adresa i veličina bafera za prijem poruke. Slanje poruke je asinhrono, dok je prijem blokirajući. U slučaju bilo kakave greške, svi ovi pozivi vraćaju neku negativnu vrednost.

Za sve operacije sa fajl sistemom zadužen je poseban sistemski proces-osluškivač (*listener*) čiji je identifikator dat konstantom `FILE_LISTENER`. Korisnički proces zadaje operaciju sa fajlom slanjem poruke ovom procesu kroz opisani sistem međuprocesne komunikacije. Poruka je niz znakova koji ima format „komandne linije“ u kojoj prva reč označava operaciju, a zatim slede argumenti razdvojeni zarezima. Na primer, za operaciju otvaranja fajla, poruka treba da ima sledeći format:

```
open <pid>,<fname>,<mode>
```

Prvi argument je celobrojna vrednost identifikatora procesa koji zahteva operaciju, drugi je naziv fajla, a treći je decimalna predstava broja čija binarna vrednost sadrži flegove za prava pristupa koja proces zahteva do fajla.

Kada izvrši ovu operaciju, proces zadužen za fajl sistem odgovara porukom u kojoj je decimalna predstava identifikatora otvorenog fajla (pozitivna decimalna vrednost) ili negativan kod greške.

Realizovati bibliotečnu sistemsku funkciju `fopen` za sistemski poziv otvaranja fajla. U slučaju uspeha, ona vraća celobrojni identifikator otvorenog fajla. U slučaju neke greške, ova funkcija treba da vrati neki negativan kod greške.

```
int fopen(const char* fname, int mode);
```

Rešenje:

2. (10 poena) Operativni sistem Linux

Napisati bash skript koji sortira listu ispita. Lista ispita se nalazi u tekstualnom fajlu. Ime fajla se zadaje kao prvi parametar prilikom poziva skripta. Ispite je potrebno sortirati po godini u kojoj se polažu (prvo idu ispiti iz prve godine, pa iz druge, itd.). Ispit je zadat kao jedan red u fajlu. Jedan red sadrži šifru ispita i još nekoliko drugih informacija koje se tiču ispita. Šifra ispita je jedna reč koja srži sledeća polja: statut (dve cifre), modul (jedno slovo), šifru odseka (dve cifre), godina (jedna cifra) i šifra predmeta (znakovi do kraja reči). Sortirane ispite ispisati na standardnom izlazu. Ukoliko se skript pozove sa neodgovarajućim brojem argumenata ispisati poruku o grešci i prekinuti izvršavanje.
Primer jednog ispita:
13E112RM1 Racunarske mreze 1 1 prvi deo 10.01.2015. 15:00

Rešenje:

3. (10 poena) Operativni sistem Linux

Na jeziku C/C++ koristeći mehanizam prosleđivanja poruka putem poštanskog sandučeta (*message queue*) kod operativnog sistema Linux, implementirati serverski proces koji obezbeđuje sinhronizaciju prolaska vozila kroz kružni tok. U jednom trenutku u kružnom toku najviše se može nalaziti *N* vozila. U kružni tok uliva se *M* ulica, koje su označene rednim brojem 1..M. Svako vozilo predstavljeno je jednim klijentskim procesom koji prilikom zahteva za ulazak u kružni tok poziva funkciju `enter` koja prosleđuje serverskom procesu poruku sa rednim brojem ulice iz koje zahtev dolazi. U slučaju da više vozila čeka na ulazak u kružni tok, vozila se puštaju po redosledu rednog broja ulice iz koje dolaze (u jednoj ulici samo jedno vozilo može čekati na ulazak u raskrsnicu, ostala vozila čekaju iza njega). Po izlasku iz kružnog toka vozilo to prijavljuje serverskom procesu. Parametri *M* i *N* zadaju se kao argumenti prilikom poziva programa. Nije potrebno proveravati uspešnost izvršavanja operacija nad sandučićima.

```
#define MESSAGE_Q_KEY 1
struct requestMsg {
 long mtype;  // msg type – street ID
 char msg[1]; // value not relevant
};

void enter(int streetId) {
 int requestMsgQueueId = msgget(MESSAGE_Q_KEY, IPC_CREAT | 0666);
 int responseMsgQueueId = msgget(MESSAGE_Q_KEY + 1, IPC_CREAT | 0666);
 size_t len = sizeof(char);

 //request entrance
 struct requestMsg msg;
 msg.mtype = streetId;
 msg.msg[0] = (char) 1;
 msgsnd(requestMsgQueueId, &msg, len, 0);
 msgrcv(responseMsgQueueId, &msg, len, streetId, 0);

 //enter
 sleep(1);

 //leave
 msg.mtype = M+1;
 msg.msg[0] = (char) 1;
 msgsnd(requestMsgQueueId, &msg, len, 0);

}
```

Rešenje: