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: Integralni, jul 2021.
Datum: 4. 7. 2021.

Integralni kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 2 sata. Dozvoljeno je korišćenje literature.

Zadatak 1	_____/10	*Zadatak 3*	_____/10
Zadatak 2	_____/10	*Zadatak 4*	_____/10

Ukupno: _____/40

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)

U nekom velikom binarnom fajlu zapisane su dve velike kvadratne matrice *a* i *b* elemenata tipa `int`, dimenzije $N \times N$ i obezbeđen prostor za smeštanje još jedne takve matrice *c* u koju treba upisati rezultat množenja $c = a \times b$. Korišćenjem principa preklopa (*overlay*), implementirati funkciju `mat_mul()` koja realizuje ovo množenje. *N* je velik broj tipa `size_t`, pa u memoriju ne mogu da stanu cele navedene matrice, već samo po jedna vrsta ili kolona svake od njih. Na raspolaganju su sledeće implementirane funkcije:

- `void load_arow (int (*)[], size_t i)`: u niz na koji ukazuje prvi parametar iz fajla učitava vrstu matrice *a* datu drugim parametrom;
- `void load_bcol (int (*)[], size_t i)`: u niz na koji ukazuje prvi parametar iz fajla učitava kolonu matrice *b* datu drugim parametrom;
- `void store_crow (int (*)[], size_t i)`: iz niza na koji ukazuje prvi parametar u fajl upisuje vrstu matrice *c* datu drugim parametrom.

Ne treba otvarati dati fajl (on je već otvoren) niti ga zatvarati (zatvoriće ga pozivalac). Sve eventualne greške obrađuju date implementirane funkcije.

Rešenje:

2. (10 poena)

Korišćenjem sistemskog poziva *fork* i POSIX semafora za sinhronizaciju napisati program koji, kad se nad njim pokrene proces, pokrene jedno svoje dete, a potom ta dva procesa strogo naizmenično na svoj standardni izlaz ispisuju po jedno „A“ (proces-roditelj), odnosno „B“ (proces-dete) tačno po 10 puta, nakon čega se završavaju. Za sistemski poziv *fork* pretpostavlja se sledeće:

- proces-dete nasleđuje standardni izlaz od roditelja, a operativni sistem obezbeđuje međusobno isključenje ispisa na isti standardni izlaz;
- proces-dete nasleđuje deskriptore semafora koje je proces-roditelj otvorio, tako da isti deskriptori ukazuju na isti semafor u sistemu koji je deljen između roditelja i deteta; oba procesa moraju da zatvore semafor kada im više nije potreban.

Rešenje:

3. (10 poena)

U računaru postoji ulazni uređaj koji prihvata pakete sa neke komunikacione veze. Paket je veličine `PACKET_SIZE` (u `sizoef(char)`). Kada paket stigne u interni hardverski bafer uređaja, uređaj generiše prekid na koji je vezana rutina `packetArrived` kernela. Kernel treba da prebaci paket u svoj interni memorijski kružni bafer `packetBuffer` koji ima mesta za `BUFFER_SIZE` paketa. Ostatak kernela te pakete iz kružnog bafera kernela isporučuje dalje kome treba (procesima). Kada prebaci paket u svoj kružni bafer, kernel treba to da signalizira uređaju upisom konstante `IO_COMPLETE` u upravljački registar uređaja, kako bi ovaj znao da može da prihvati naredni paket u svoj interni bafer. Ukoliko je kružni bafer kernela pun, kernel treba uređaju da javi da paket mora da se odbaci upisom konstante `IO_REJECT` u upravljački registar uređaja. Interni hardverski bafer uređaja je veličine jednog paketa i preslikan je u memorijski prostor počev od adrese `ioData`. Date su deklaracije pokazivača preko kojih se može pristupiti registrima kontrolera ulaznog uređaja, kao i deklaracije za kružni bafer:

```
typedef volatile unsigned int REG;
REG* ioCtrl = ...;   // Input device control register
volatile char* ioData = ...; // Input device packet buffer

extern const size_t PACKET_SIZE, BUFFER_SIZE;
extern char packetBuffer[BUFFER_SIZE*PACKET_SIZE];
extern char *pbHead, *pbTail;

interrupt void packetArrived ();
```

Glava `pbHead` ukazuje na prvi znak paketa u kružnom baferu koji je na redu za prenos u ostatak kernela, a rep `pbTail` ukazuje na prvi znak prvog slobodnog mesta za prijem novog paketa u kružni bafer. Tokom izvršavanja prekidnih rutina drugi prekidi su maskirani i nema uporednog izvršavanja ostalog koda kernela.

Realizovati prekidnu rutinu `packetArrived`.

Rešenje:

4. (10 poena)

Neki fajl sistem primenjuje kombinovanu tehniku indeksirane alokacije sadržaja fajla. U FCB fajla polje `singleIndex` predstavlja direktni indeks kao niz od `SingleIndexSize` elemenata, pri čemu svaki element sadrži broj fizičkog bloka sa sadržajem fajla (ili 0, ako je neiskorišćen). Ako veličina sadržaja fajla preraste veličinu podržanu ovim indeksom, naredni blokovi sadržaja fajla (preko ove veličine) indeksiraju se indeksom u dva nivoa. Za te potrebe, polje `dblIndex` u FCB sadrži broj bloka na disku u kom je indeks drugog nivoa. Taj blok sa indeksom drugog nivoa sadrži `DblIndexSize` ulaza sa brojevima blokova sa sadržajem fajla, odnosno 0 ako ulaz nije iskorišćen. Polje `size` u FCB sadrži veličinu fajla u bajtovima. Veličina bloka u bajtovima je `BLOCK_SIZE`. Svi ulazi u indeksu prvog nivoa i indeksu drugog nivoa (ako postoji) su sigurno postavljeni na validne vrednosti (broj bloka sa sadržajem ili 0).

Realizovati funkciju `extendFile` koja proširuje sadržaj fajla čiji je FCB dat za jedan blok:

```
int extendFile (FCB* fcb);
```

Ova funkcija treba da vrati 0 u slučaju uspeha, a negativnu vrednost u slučaju greške. Ona ne treba da menja polje `size` u FCB (to će raditi pozivalac). Na raspolaganju je funkcija koja vraća pokazivač na sadržaj bloka na disku sa datim brojem, učitanog u keš:

```
void* getBlock (PBlock blkNo);
```

kao i funkcija koja zauzima jedan slobodan blok i vraća njegov broj ili 0 ako takvog nema:

```
PBlock allocateBlock ();
```

Rešenje: