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: Treći, jun 2023.
Datum: 11. 6. 2023.

Treći kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje 90 minuta. 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 ulazno-izlaznom podsistemu nekog operativnog sistema zahtevi za operacijama sa diskom predstavljaju se instancama strukture `DiskRequest`. Jedan zahtev odnosi se na prenos `blockCount` susednih blokova počev od bloka broj `startBlockNo`, a prenos se vrši sa baferom `buffer` i u smeru koji zadaje polje `dir`.

```
struct DiskRequest {
  uint32  blockCount;
  uint32  startBlockNo;
  uint32* buffer;
  enum Dir : uint32 {in=0, out=1};
  Dir dir;
  ... // Other details irrelevant here
};
```

Za svaki fizički uređaj formira se poseban red `RequestQueue` ovakvih zahteva. Svaki takav red opslužuje po jedna interna nit jezgra koja izvršava funkciju `diskDriver` prikazanu dole. Ova funkcija uzima jedan po jedan zahtev iz reda i svaki zahtev zadaje drajveru uređaja `dd` pozivom njegove operacije `startTransfer`. Potom se blokira na semaforu `semComplete` koji drajver uređaja treba da signalizira kada zahtev bude opslužen, a onda dalje obaveštava podnosioca zahteva (izostavljeni detalji).

```
void diskDriver (RequestQueue* rque, IBlockDeviceDriver* dd) {
  while (true) {
    DiskRequest* req = rque->getRequest();// Blocks until a request arrives
    dd->startTransfer(req);
    semComplete->wait();
    ... // Notify the request initiator
  }
}
```

Interfejs drajvera uređaja `IBlockDeviceDriver` dat je dole. Kada se drajver instalira, jezgro poziva njegovu operaciju `init` za inicijalizaciju samog drajvera, zadajući semafor koji treba signalizirati nakon svakog opsluženog zahteva (`semComplete` pomenut gore).

```
class IBlockDeviceDriver {
public:
  virtual int  init (Semaphore* complete) = 0;
  virtual void startTransfer (DiskRequest*) = 0;
};
```

Tokom inicijalizacije u svojoj operaciji `init` drajver može da poziva sledeće operacije jezgra:

- `uint32* requestDMAChannel()`: zahteva od jezgra zauzimanje jednog slobodnog DMA kontrolera (kanala) za svoje potrebe; jezgro dodeljuje takav kontroler na upotrebu drajveru, ukoliko ga ima, i vraća adresu početka regiona memorijski mapiranih upravljačkih registara dodeljenog DMA kontrolera; ukoliko slobodnog DMA kontrolera nema, vraća *null*;
- `uint32 requestIVTEntry(void (*transferComplete)(void*),void* ptr))`: od jezgra traži zauzeće i inicijalizaciju jednog slobodnog ulaza u IVT; ukoliko ne uspe, vraća negativnu vrednost; ukoliko uspe, inicijalizuje taj ulaz u IVT adresom funkcije zadate prvim parametrom i vraća taj broj ulaza; jezgro obezbeđuje da se prekidna rutina poziva sa parametrom `ptr` (prekidna rutina na koju je usmeren jedan ulaz u IVT poziva se uvek sa istim tim parametrom).

Ukoliko inicijalizacija ne uspe, funkcija `init` treba da vrati negativnu vrednost, 0 za uspeh.
Svaki DMA kontroler obavlja prenos sa samo jednim diskom i ima dva 32-bitna upravljačka registra u svom regionu memorijski mapiranih registara. Prvi registar nalazi se na pomeraju nula od početka regiona, a prilikom inicijalizacije DMA kontrolera u ovaj registar potrebno je

upisati broj ulaza u IVT koji će ovaj DMA kontroler koristiti prilikom prekida koji generiše kada završi svaku zadatu operaciju. Drugi registar nalazi se na pomeraju 4 (adresibilna jedinica je bajt, adrese su 32-bitne) od početka regiona, a jedna operacija prenosa DMA kontroleru se zadaje sukcesivnim upisima u ovaj isti upravljački registar na sledeći način (nije potrebno čekati na dozvolu za sledeći upis, ovi upisi mogu da idu odmah jedan iza drugog): najpre je u ovaj registar potrebno upisati adresu bafera, zatim broj prvog bloka na disku, zatim broj susednih blokova koje treba preneti i konačno smer prenosa (0 ili 1). Ovaj poslednji upis ujedno i pokreće prenos. Po završetku prenosa DMA kontroler generiše prekid.

Realizovati u potpunosti klasu `DiskDeviceDriver` koja implementira interfejs `IBlockDeviceDriver`, a koja realizuje drajver za prenos sa disk uređajem korišćenjem DMA kontrolera. Jedan objekat ove klase treba da bude zadužen za jedan fizički disk uređaj.

Rešenje:

2. (10 poena) Fajl sistem

U sistemima nalik sistemu Unix cevovod (*pipe*) se formira sistemskim pozivom *pipe*:

```
#include <unistd.h>
int pipe (int pipefd[2]);
```

Ovaj sistemski poziv pravi cevovod i u `pipefd[0]` upisuje deskriptor fajla (*file descriptor*) odredišne strane otvorenog cevovoda (strana za čitanje), a u `pipefd[1]` upisuje desktriptor fajla izvorišne strane tog cevovoda (strana za upis). U slučaju greške, ovaj poziv vraća negativnu vrednost, a u slučaju uspeha vraća 0. Dati su i sistemski pozivi za čitanje i upis:

```
ssize_t read  (int fd, void *buf, size_t count);
ssize_t write (int fd, void *buf, size_t count);
```

Ovi pozivi prenose najviše `count` znakova sa fajlom koga identifikuje `fd` iz bafera ili u bafer na koga ukazuje `buf`, a vraćaju stvarno prenesen broj znakova.

Korišćenjem ovih, kao i drugih potrebnih sistemskih poziva i bibliotečnih funkcija (*fork*, *exit*, *close*, *putchar*, *fprintf*), napisati program koji, kada se nad njim pokrene proces, formira jedan cevovod kojim komunicira sa procesom detetom koga pokreće. Nakon toga, proces roditelj kroz cevovod procesu detetu šalje niz znakova dat svojim argumentom komandne linije `argv[1]` i završen znakom `'\0'`, a proces dete dobijeni niz znakova ispisuje na svoj standardni izlaz. Nakon što uradi navedeno, svaki od procesa se gasi. Greške obraditi ispisom na `stderr` i gašenjem procesa.

Rešenje:

3. (10 poena) Fajl sistem

U implementaciji nekog FAT fajl sistema ceo FAT keširan je u memoriji u nizu `fat`:

```
extern uint32 fat[];
extern uint32 freeHead, freeCount;
```

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 i polje `size` koje sadrži veličinu sadržaja fajla u bajtovima. Slobodni blokovi su ulančani u jednostruku listu čija glava je u promenljivoj `freeHead`, dok ukupan broj slobodnih blokova čuva promenljiva `freeCount`. Implementirati sledeću funkciju koja treba da obriše sadržaj datog fajla:

```
void truncate (FCB* fcb);
```

Rešenje: