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, septembar 2/2025.
Datum: 5. 10. 2025.

Kolokvijum iz Operativnih sistema 1

Kandidat: _____

Broj indeksa: _____ *E-mail:*_____

Kolokvijum traje dva 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 sistemu sa straničnom organizacijom memorije adresibilna jedinica je bajt, virtuelni adresni prostor je veličine 1 MB, a veličina stranice je 4 KB. Dat je spisak početnih (virtuelnih) adresa, veličina (obe vrednosti su zapisane heksadecimalno) i vrsta logičkih segmenata (regiona) koje je alocirao neki proces.

Adresa segmenta (hex)	*Veličina (hex)*	*Vrsta segmenta*
0	2760	instrukcije
4000	FF0	konstantni podaci inicijalizovani statički
5000	3E68	instrukcije
17000	170B	promenljivi podaci
EE000	2000	stek

a)(5) U datu tabelu upisati parametre svih stranica koje je operativni sistem organizovao u PMT (sve ulaze u PMT koji nisu *null*) za ovaj proces, po rastućem redosledu broja stranice. Broj stranice zapisati heksadecimalno, a bite prava pristupa *RWX* binarno tim redom. (Broj redova u datoj tabeli ne mora da odgovara broju stranica koje treba upisati; eventualni višak redova date tabele ostaviti prazne.)

Stranica # (hex)	*RWX (bin)*

b)(3) Ako je PMT organizovan u dva nivoa, a PMT prvog nivoa ima 16 ulaza, koliko će PMT drugog nivoa biti alocirano za ovaj proces i koliko ulaza ima svaki od njih?

Odgovor: _____

c)(2) Kada bi PMT bio organizovan u samo jednom nivou, koliko ulaza bi imao PMT?

Odgovor (*hex*): _____

2. (10 poena)

U školskom jezgru implementirani su sledeći elementi:

- tip `Timing::Time`, iza kog se krije dovoljno velik neoznačen celobrojni tip čije vrednosti predstavaljaju vremenske intervale izražene u malim jedinicama vremena;
- operacija `Timing::now()` koja vraća tekuće vreme tipa `Time` kao vreme proteklo od nekog referentnog trenutka u prošlosti;
- operacija `Timing::sleep(Time)` koja uspavljuje tekuću nit do trenutka koji je zadat argumentom (trenutak se zadaje kao interval protekao od referentnog trenutka).

Korišćenjem ovih stvari, implementirati klasu `PeriodicThread` čiji je interfejs dat dole i koja treba da implementira periodične niti. Perioda aktivacije niti (kao interval) zadaje se prvim argumentom konstruktora. Drugi argument konstruktora je opcioni i ako je različit od nule, zadaje trenutak u kom periodičnu nit treba pokrenuti nakon poziva operacije `start` (a ne odmah po pozivu operacije `start`). Svake periode treba pozvati apstraktnu operaciju `activate` koju izvedene klase treba da redefinišu i u njoj implementiraju aktivnost niti u svakoj periodičnoj aktivaciji. Nakon poziva operacije `stop`, nit treba da se završi prilikom prve naredne periodične aktivacije.

```
using Timing::Time;

class PeriodicThread : public Thread {
public:
  PeriodicThread (Time period, Time start = 0);

  void stop ();
  virtual void activate () = 0;
  ...
};
```

Rešenje:

3. (10 poena)

Dole je dat interfejs drajvera blokovski orijentisanog uređaja („diska“). Drajver obavlja ulazno-izlazne operacije sa diskom pomoću DMA kontrolera. Operacijom `setIVTE` drajveru se zadaje dodeljen broj ulaza u IVT preko kog će DMA signalizirati završetak zadate operacije; drajver ovaj broj upisuje u odgovarajući registar DMA kontrolera. Operacija sa DMA pokreće se pozivom operacije `start` sa zadatim brojem bloka na disku `blkNo`, za prenos bloka podataka na zadatoj adresi `buffer` i za odgovarajući smer (`rdwr`, 0 za čitanje, 1 za upis). Nakon završenog prenosa, DMA kontroler generiše prekid sa zadatim brojem ulaza u IVT, a nakon toga status završene operacije može se očitati pozivom operacije `getStatus`.

Mehanizam prekida omogućava da se u IVT, pored pokazivača na prekidnu rutinu, zada i parametar tipa `void*` koji odgovara svakom ulazu. Taj parametar se dostavlja prekidnoj rutini prilikom obrade prekida u tom ulazu. Ulaz u IVT postavlja se na zadatu rutinu `isr` operacijom `Interrupts::initIVT`.

Dole je data implementacija klase `BlockDevice`, zajedno sa odgovarajućom prekidnom rutinom, koja uporednim korisničkim nitima pruža usluge prenosa sa blokovskim uređajem. Pri inicijalizaciji, objektu ove klase dostavlja se broj ulaza u IVT koji je dodeljen datom uređaju, kao i drajver tog uređaja. Niti zadaju operacije sa diskom pozivom operacije `perform`. Ova operacija obavlja se jedna po jedna, redom kako su korisničke niti nju pozvale.

Prepraviti implementaciju ove klase, bez izmene u interfejsu prema korisničkim nitima, tako da se zahtevi korisničkih niti stavljaju u ograničeni bafer, a ne odmah sinhrono obrađuju redom. Nit koja je postavila zahtev treba da se suspenduje dok se zahtev ne izvrši. Posebna nit uzima ove zahteve iz bafera jedan po jedan i za njih pokreće operacije sa drajverom uređaja, što omogućava obradu zahteva redosledom koji nije isti kao onaj kojim su u bafer stavljeni. Ograničeni bafer implementiran je gotovom šablonskom klasom `BoundedBuffer<typename T, int capacity>`, parametrizovanom tipom elementa koji se stavlja u bafer i njegovim kapacitetom, a čiji je interfejs dat na predavanjima. Sinhronizaciju obavljati semaforima koje implementira klasa `Semaphore` sa interfejsom kao u školskom jezgru, a niti izvođenjem iz klase `Thread` kao u školskom jezgru.

```
class DeviceDriver {
public:
  virtual void setIVTE (IVTNo ivte);
  virtual void start (BlkNo blkNo, void* buffer, int rdwr);
  virtual int  getStatus ();
};

void Interrupts::initIVT (IVTNo ivte, void(*isr)(void*), void*);

class BlockDevice {
public:
  BlockDevice (IVTNo ivte, BlkDeviceDriver* drv);
  static const int RD = 0, WR = 1;
  int perform (BlkNo blkNo, void* buffer, int rdwr);

private:
  friend void blkDevISR (void*);
  Semaphore queue, eop;
  BlkDeviceDriver* myDrv;
};

interrupt void blkDevISR (void* dev) {
  BlockDevice* d = (BlockDevice*)dev;
  d->eop.signal();
}
```

```
BlockDevice::BlockDevice (IVTNo ivte, BlkDeviceDriver* drv)
  : queue(1), eop(0), myDrv(drv) {
  Interrupts::initIVT(ivte, blkDevISR, this);
  myDrv->setIVTE(ivte);
}

int BlockDevice::perform (BlkNo blkNo, void* buffer, int rdwr) {
  this->queue.wait();
  myDrv->start(blkNo,buffer,rdwr);
  this->eop.wait();
  int status = myDrv->getStatus();
  this->queue.signal();
  return status;
}
```

Rešenje:

4. (10 poena)

Fajl sistem FAT16 koji je koristio MS DOS svaki ulaz u direktorijumu predstavlja strukturom `FATDirectoryEntry` koja zauzima tačno 32 bajta i u kojoj su upisani atributi fajla ili poddirektorijuma. Spisak elemenata direktorijuma zapisuje se kao sadržaj bilo kog fajla. Ovaj spisak je prost niz ovakvih struktura koje se redom dodaju u niz kako se elementi dodaju u direktorijum, pri čemu se za obrisane tj. slobodne ulaze prvi bajt ove strukture postavlja na posebnu vrednost `0xE5`. (Ova struktura zapravo predstavlja FCB koji je ugrađen u sam zapis sadržaja direktorijuma.) Jedan bit u polju `attr` ukazuje na to da li ulaz predstavlja fajl ili poddirektorijum. Polje `firstCluster` u ovoj strukturi ukazuje na prvi blok sa zapisom sadržaja fajla odnosno poddirektorijuma.

```
typedef struct {
    char     name[8];         // File name (padded with spaces)
    char     ext[3];          // File extension (padded with spaces)
    uint8_t  attr;            // File attributes
    uint8_t  reserved;        // Reserved (used for Windows NT or VFAT)
    uint8_t  creationTimeTenth; // Creation time in tenths of a second
    uint16_t creationTime;    // Creation time
    uint16_t creationDate;    // Creation date
    uint16_t lastAccessDate;  // Last access date
    uint16_t highCluster;     // Unused; always 0 for FAT16
    uint16_t writeTime;       // Last write time
    uint16_t writeDate;       // Last write date
    uint16_t firstCluster;    // First cluster
    uint32_t fileSize;        // File size in bytes
} FATDirectoryEntry;
```

Prostor na particiji organizuje se u tri dela:

- na početku particije je FAT koji zauzima određen broj blokova;
- iza toga se nalazi zapis sadržaja korenog direktorijuma koji čini niz od 512 struktura `FATDirectoryEntry`, pa zauzima određen fiksan broj blokova;
- iza toga se nalaze blokovi u koje se smešta sadržaj fajlova i direktorijuma.

Ako se pretpostavi da je ceo FAT učitan i keširan u operativnoj memoriji, a da osim njega nijedan više blok korenog direktorijuma ili sadržaja fajlova nije učitan (keširan), kao i da svi pomenuti direktorijumi i fajlovi postoje, koliko blokova sa diska mora da bude učitano da bi se odredio broj prvog bloka sa sadržajem fajla `file.txt` (data staza sadrži *n* direktorijuma)? Precizno obrazložiti odgovor.

`\dir_1\dir_2\ ... \dir_n\file.txt`

a)(5) ako se pretpostavi da su svi navedeni direktorijumi i dati fajl jedini elementi svojih roditeljskih direktorijuma i nalaze se u njegovom prvom ulazu:

Odgovor: _____

Objašnjenje:

b)(5) ako se pretpostavi da su svi navedeni direktorijumi i dati fajl smešteni u 20. ulaz svojih roditeljskih direktorijuma, a blok je veličine 512 bajtova (klaster je jedan blok):

Odgovor: _____

Objašnjenje: