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, Oktobar1/2025.
Datum: 19. 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)

Neki sistem vodi evidenciju alociranih (deklarisanih) logičkih segmenata (regiona) za svaki proces u jednostruko ulančanoj listi elemenata tipa `RegionDesc` uređenih po početnoj virtuelnoj adresi segmenta (polje `addr`). Svaki logički segment opisuje jedan deskriptor tipa `RegionDesc` u kom polje `size` sadrži veličinu segmenta. Ova lista je uvek neprazna, jer kernel za svaki proces, prilikom njegovog pokretanja, alocira najmanje jedan logički segment na najnižim adresama virtuelnog adresnog prostora (počev od adrese 0) koji preslikava u svoj memorijski prostor.

```
struct RegionDesc {
  byte* addr; size_t size;
  RegionDesc* next;
};

void* createRegion (RegionDesc* phead, size_t sz);
```

Implementirati funkciju `createRegion` koja se koristi u implementaciji sistemskog poziva `mmap` sa parametrom `start_addr` jednakim `NULL`. Ova funkcija treba da alocira (deklariše) nov logički segment tražene veličine (`sz`) na mestu u virtuelnom adresnom prostoru procesa u kom ima dovoljno prostora tražene veličine tako da se ne preklapa sa već alociranim segmentima, po *best-fit* algoritmu, i uključi ga u evidenciju segmenata procesa. U slučaju uspeha funkcija vraća početnu adresu alociranog segmenta, a u slučaju neuspeha vraća *null*. Najveća adresa u virtuelnom adresnom prostoru je `MAX_VADDR`. Argument `phead` je glava liste deskriptora segmenata datog procesa. Tip `byte` predstavlja celobrojni tip veličine jedne adresibilne jedinice. Ne treba poravnavati adrese i veličine segmenata. Dinamičku alokaciju prostora za potrebe struktura jezgra radi funkcija `kmalloc` koja ima isti potpis i dejstvo kao standardna C funkcija `malloc`.

Rešenje:

2. (10 poena)

Korišćenjem semafora u školskom jezgru, na jeziku C++ implementirati šablonsku klasu `BoundedBuffer` koja implementira ograničeni bafer i čiji je interfejs dat dole. Više proizvođača stavlja u bafer pakete veličine `PackageSize` elemenata tipa `T`, dok više potrošača iz bafera uzima pojedinačne elemente tipa `T`. Kapacitet bafera je `NumOfPkgs` paketa veličine `PackageSize` elemenata tipa `T`. Omogućiti uporedne aktivnosti proizvođača i potrošača (dok jedan proizvođač uzima element, neki potrošač može da stavlja element, ako su potrebni uslovi zadovoljeni).

```
template<typename T, int PackageSize, int NumOfPkgs>
class BoundedBuffer {
public:
  BoundedBuffer ();
  void append (T pkg[]); // pkg is an rray of size PackageSize
  T take ();
};
```

Rešenje:

3. (10 poena)

Neki program obilazi i obrađuje jedno ogromno kompletno binarno stablo sa n nivoa i tačno 2^n-1 čvorova, tako što obilazak počinje od korena i svaki čvor obilazi i čita samo po jednom. Stablo je zapisano kao niz čvorova poput strukture hipa (*heap*): koreni čvor je u elementu niza sa indeksom 0, levo dete čvora sa indeksom i je u elementu sa indeksom $2i$+1, a desno dete je u elementu sa indeksom $2i$+2. Ovaj niz je redom po indeksima zapisan u fajl koji je smešten na disku sa blokom u koji staju tačno dva susedna elementa niza (elementi 0 i 1 su u jednom bloku, elementi 2 i 3 su u sledećem itd). Ako sistem za svaki fajl kešira uvek jedan i samo jedan blok sa njegovim sadržajem, izračunati koliko je operacija učitavanja bloka sa diska potrebno izvršiti tokom obrade celog ovog stabla ako je obilazak u sledećem redosledu:

a)(5) prefiksno, odozgo nadole po dubini (*pre-order top-down depth-first*);

b)(5) odozgo nadole po širini (*top-down breadth-first*).

Odgovor precizno obrazložiti.

Rešenje:

4. (10 poena)

Implementirati funkciju `setAccessBits` koja se koristi u implementaciji sistemskog programa *chmod*; ona postavlja tri bita za prava pristupa do fajla za neku od klasa korisnika (vlasnika, grupu ili ostale).

Parametar `oldBits` u najniža tri bita sadrži dosadašnja prava pristupa (*r* u bitu 2, *w* u bitu 1 i *x* u bitu 0). Novopostavljene vrednosti treba upisati u najniža tri bita parametra `newBits`. Parametar `str` sadrži barem jedan i najviše 4 znaka. U `str[0]` je uvek jedan od znakova `=`, `+` ili `-`, sa značenjem kao u sistemskom pozivu *chmod*. U preostala tri znaka mogu (ali ne moraju) da se nađu velika ili mala slova `r`, `w` ili `x` u proizvoljnom redosledu; sve ostale znakove koji se tu eventualno pojave treba ignorisati, a navedeni znakovi smeju i da se ponavljaju.

```
void setAccessBits (const char str[4], short oldBits, short& newBits);
```

Rešenje: